An Effective Methodology to Traceback DDoS Attackers

LAW, Kwok Tai

A Thesis Submitted in Partial Fulfillment
of the Requirements for the Degree of
Master of Philosophy
in
Computer Science and Engineering

June, 2003

Abstract

Recently, there is an urgent need on how one can effectively defend against distributed denial-of-service (DDoS) attack to many well established web sites. Due to the increase of sophistication and severity of DDoS attack, the system administrator of a victim web site needs to quickly and accurately identify the potential attackers and eliminate the attack traffics. Our work is based on a probabilistic marking algorithm in which an attack graph can be constructed by a victim site. We extend the concept further such that one can quickly and efficiently deduce the intensity of the "local traffic" generated at each router in the attack graph based on the volume of received marked packets at the victim site. Given the intensities of these local traffic rates, one can perform ranking and identify which network domains generate the most traffic that consume most of the victim's resource. We provide a theoretical framework to determine the minimum stable time t_{min}, which is the minimum time to accurately determine the local traffic rate of every participating router in the attack graph. Experiments are carried out to illustrate that we can accurately determine the minimum stable time t_{min} under various threshold parameters, different network diameters, attack traffic distributions and various network traffic patterns.

摘要

最近分佈式拒絕服務程式攻擊了很多良好建立的網站，我們急切需要一個有效防衛的方法。由於分佈式拒絕服務程式的攻擊越來越嚴重而且增長迅速，被攻擊的網站的系統管理員有需要迅速而準確地辨認潛在的攻擊者和消除他們的攻擊。我們的研究是根據隨機網路標記的算法繪製出攻擊圖表。我們進一步拓展這個概念，使被攻擊的網站能根據所收到信息包裏的網路標記，有效迅速地測算出本地信息流量的密度。如果有本地信息流量的密度，被攻擊的網站能從密度的大小，辨認出那一個網路領域產生最多的信息包消耗網站資源。我們提供一個理論框架以找出本地信息流量的最小化準確測算時間，這是在攻擊圖表中，以最小時間準確地測算出每台路由器的本地信息流量。實驗結果證明，在不同的測算參數，不同的網路直徑，不同的攻擊分佈和不同的網路交通模式之下，我們能正確地找出最小化準確測算時間。

ACKNOWLEDGMENTS

I wish to express my gratitude to my supervisor Professor John C.S. Lui for giving continuous support and guidance in research. I would like to thank Professor Irwin King and Professor K.H. Wong for giving many helpful advice. I would also like to thank Chan Chi Bun and Lau Chi Fai for their support in the software programmable router implementations.

Contents

List of Figures

List of Tables

Chapter 1

Introduction to Network Security via Efficient IP Traceback

1.1 Motivation

One may argue that the important factors which contribute to the success of the Internet are the open standard and well-known interfaces (e.g., sockets API) which allow many users to quickly deploy new applications and services. However, also because of this open architecture, it allows people to create viruses and network attacks to the Internet community. A prime example is the distributed denial-of-service (DDoS) attack. DDoS attack is a pressing problem on the Internet. Well established commercial sites such as Yahoo, Amazon and eBay were being attacked and were out of service for many hours due to the flooding-based DDoS attack in February 2000 [ddo00]. The attackers generated a large number of packets to overwhelm the bandwidth and the system resources of these sites. In recent years, DDoS attacks have increased in frequency, sophistication and severity. A recent study by the Computer Emergency Response Team (CERT) has indicated that the number of DDoS attacks has increased by 50% per year [How98]. More alarming is the fact that

many sophisticated DDoS attack tools, such as Tribal flood network (TBN), TFN2K and Trinoo can be easily downloaded from the Internet so users of these attack tools can simply launch a large-scale site attack with a few key strokes.

1.2 DDoS Attacker Traceback Problem

The major difficulty to defend against a DDoS attack is that attackers often use *fake*, or *spoofed* IP addresses as their source IP addresses. Therefore, attackers can easily disguise themselves as some other hosts on the Internet. Because of the stateless nature of the Internet (e.g., core router does not store any path or source information), it is difficult to determine or to trace the originating source of these attackers' packets and thereby locating the potential locations of these DDoS attackers. This is known as the DDoS attacker *traceback* problem.

One intrinsic characteristic of a DDoS attack is that source addresses of packets are usually spoofed. To avoid these spoofed packets into the Internet, an ISP network administrator can implement services such as ingress filtering [FS98], for example, refusing packet forwarding service whenever the packet's source IP address is not within the ISP's administrative domain. If these packets are dropped, they will not be able to enter the Internet core and thereby reducing the possibility of a DDoS attack. However, ingress filtering required edge routers to have sufficient processing power, not only to inspect the packet's destination IP address for normal packet forwarding service, but also need to inspect the source IP address and determine whether it is a legitimate or illegitimate address. Another major problem about the ingress filtering approach is that this technique is only effective if there is a *"widespread deployment"* in the networking community (e.g., many ISPs are willing to deploy this service). Moreover, even with the enabling of ingress filtering service, attackers can still forge the source IP addresses as other hosts

which are within the edge router's local network domains. Researchers also proposed other method to counter the DDoS attack, for example the input debugging technique [Sto00], which requires cooperation between system administrators of different ISPs. The limitation of this approach is that it may not be able to trace the DDoS attackers in real-time or in the midst of a DDoS attack (since not all administrators of all ISPs are working at the same time). Other approaches such as controlled flooding [BC00], which generates many additional packets to the network (which can be viewed as another form of DDoS attack), or network logging [SPS+01], which requires additional storage and computational overhead of the participating routers. All in all, the above approaches have performance problems and significant deployment difficulties.

Recently, a number of researchers [SWKA00, PL01, SP01] proposed probabilistic edge marking algorithm to traceback the locations of the attackers. Each participating router will probabilistically mark packets targeted to a victim site. Based on the received marked packets, the victim can construct an attack graph which depicts the traversed paths of all received packets to the victim site. This implies that the attack graph consists of the traversed paths of attack packets, as well as the traversed paths of regular packets. Therefore, one major shortcoming of this approach is that it does not identify and locate the *potential attackers*. In this work, we complement and enhance the probabilistic marking method by identifying the locations of these attackers under the flooding-based DDoS attack. The contributions of our work are:

- We propose an effective method so that the victim site can deduce the *local traffic rates* of all participating routers in the attack graph.

- We provide a theoretical framework to determine the minimum stable time t_{min}, which is the *minimum* time required to accurately determine the local traffic rates of all participating routers. Note that the lower the value of t_{min} implies that the system administrators can determine the

locations of attackers earlier.

- The traceback methodology is effective for a general network topology.

1.3 Document Roadmap

The outline of the thesis is as follows. In Chapter 2, we provide the necessary background of the probabilistic marking algorithm. In Chapter 3, we present the traceback method as well as how we can eliminate potential attackers for a simple but illustrative linear network topology. In Chapter 4, we extend our traceback methodology to a general network topology. Simulations and experiments are presented in Chapter 5 and 6 to illustrate the effectiveness of our traceback methodology. Related work is presented in Chapter 7. Lastly, conclusion is given in Chapter 8.

Chapter 2

Background

IP traceback is an approach to determine the source of an attack when a distributed denial-of-service (DDoS) attack occurs. To accomplish this, we assume that routers will provide additional *packet marking* functions besides the basic packet routing/forwarding service. In particular, whenever a packet passes through a router, the packet will be marked, either deterministically or probabilistically. The marked packets are filled with the information of their respective traversed paths. Upon receiving these marked packets, a victim site, denoted by $\mathcal{V}$, can use the marking information to create an attack graph. Based on the attack graph and the received packets, a victim site $\mathcal{V}$ trace the attackers back towards the sources (e.g., the originated router of the attack traffic). In the following, we first present the necessary background of the probabilistic edge marking algorithm [SWKA00], which is used by a victim site $\mathcal{V}$ to create an attack graph. Given the attack graph, we then present the methodology to estimate the local traffic rate of each router in the attack graph as well as the minimum time it takes to locate the potential attackers.

2.1 Probabilistic Edge Marking Algorithm

To provide IP traceback feature, one can implement this service by allocating enough space in an IP packet header so that one can use this space to record

the traversed path of a packet. For example, each router, besides performing the normal packet routing and forwarding functions, also records or appends its own IP at the packet's header so that when a victim site $\mathcal{V}$ receives a marked packet, it can examine the packet's header and obtain the complete traversed path information of the marked packet. However, the major problem about this simple approach is that the length of a traversed path (e.g., number of hops) of a packet is not fixed. For example, a packet to a victim site $\mathcal{V}$ may traverse only two hops while another packet may traverse 15 hops. Therefore, it is impossible to pre-allocate sufficient amount of space in the packet's header in a prior fashion. Another technical difficulty of recording the complete traversed path information of each packet to the victim site $\mathcal{V}$ is that an attacker can potentially manipulate this path information and fill in with false router's identification so as to mislead the victim site $\mathcal{V}$ in the path analysis.

2.1.1 Probabilistic Edge Marking Procedure

Recently, probabilistic edge marking algorithm was proposed by Savage [SWKA00] and others [PL01, SP01]. The idea is that instead of recording the complete traversed path information of a packet, one only needs to record a traversed *edge* from the attacker to the victim site $\mathcal{V}$ in a probabilistic fashion. Under the probabilistic edge marking algorithm, some unused fields in the packet's header are used to store three finite fields. These three fields are {`start, end, distance`}. The `start` and `end` fields store the IP addresses of the two routers at the end points of the marked edge while the `distance` field records the number of hops between the marked edge and the victim site $\mathcal{V}$. When a victim site $\mathcal{V}$ is under a DDoS attack, the victim site $\mathcal{V}$ will send a "`marking-request-signal`" to a set of routers (e.g., all routers which are within $d \geq 1$ hops away from the victim site $\mathcal{V}$) requesting their participations

for the probabilistic edge marking process. Each participating router will then mark each packet targeted to the victim site $\mathcal{V}$ with *probability* p. In other words, whenever an IP packet which is targeted to the victim site $\mathcal{V}$ passes through a router in the enabled router set, the router, upon deciding the out-going edge of this packet through the standard routing table lookup, also marks this out-going edge into this packet's IP header with a probability p. In this case, the router records its IP address into the `start` field and sets the value of the `distance` field to zero. If the router decides not to mark the packet, the router needs to check whether the `distance` field of the packet is equal to zero or not. If it is equal to zero, the router records its IP address in the `end` field and then increments the `distance` field by one. If the `distance` field is not equal to zero, the router simply increments the `distance` field by one. Note that the mandatory increment of the distance field is crucial because it is used to minimize the probability of spoofing a marked edge. Under this marking method, any packet generated by an attacker will have distance greater than or equal to the hop count between the victim site $\mathcal{V}$ and the attacker. Therefore, a single attacker cannot forge any edge between himself and the victim site $\mathcal{V}$. The probabilistic edge marking algorithm by each participating router in the enabled set is illustrated in Figure 2.1.

Due to the property of the probabilistic marking algorithm, each traversed edge of an attack packet will have a different probability of being marked or unmarked. Let $P_m(d)$ denote the probability that a victim site $\mathcal{V}$ will find an edge which is d hops away as a marked edge. In general, we have

$$P_m(d) = p(1-p)^d \qquad d \geq 0. \tag{2.1}$$

In other words, an edge which is d hops away from the victim site $\mathcal{V}$ will only be marked if a router which is connected to that edge decides to mark the packet and the remaining routers along the packet's traversed path decide not to mark (or reset the mark) this packet. Let $P_u(d)$ be the probability that a

Algorithm: Edge Marking Procedure at router R

```
for (each packet w targeted to the victim site V) {
 generate a random number x between [0..1);
 if (x < p) { /* router R needs to mark the pkt*/
   write R into w.start and 0 into w.distance;
 }
 else { /* router R does not need to mark */
   if (w.distance == 0) {
      write R into w.end;
   }
   increment w.distance;
 }
}
```

Figure 2.1: Probabilistic edge marking algorithm for each participating router

victim site $\mathcal{V}$ will *not* find an edge which is d hops away or closer as a marked edge. We have

$$P_u(d) = (1-p)^{d+1} \qquad d \geq 0. \tag{2.2}$$

In other words, all routers along the path to the victim site $\mathcal{V}$ decide not to mark the packet. Figure 2.2 illustrates the set of marked and unmarked edges collected by the victim site $\mathcal{V}$ under a simple linear network topology. In this example, the victim site $\mathcal{V}$ can collect four types of packets. The first three types are marked packets with edges (R_3, R_2), (R_2, R_1) and $(R_1, -)$ respectively. The last type of packet received by victim site $\mathcal{V}$ is the unmarked packet.

2.1.2 Attack Graph Construction Procedure

A victim site $\mathcal{V}$, upon receiving packets, needs to first filter out those unmarked packets (since they don't carry any information in the attack graph construction). For all the collected marked packets, the victim site $\mathcal{V}$ needs to execute

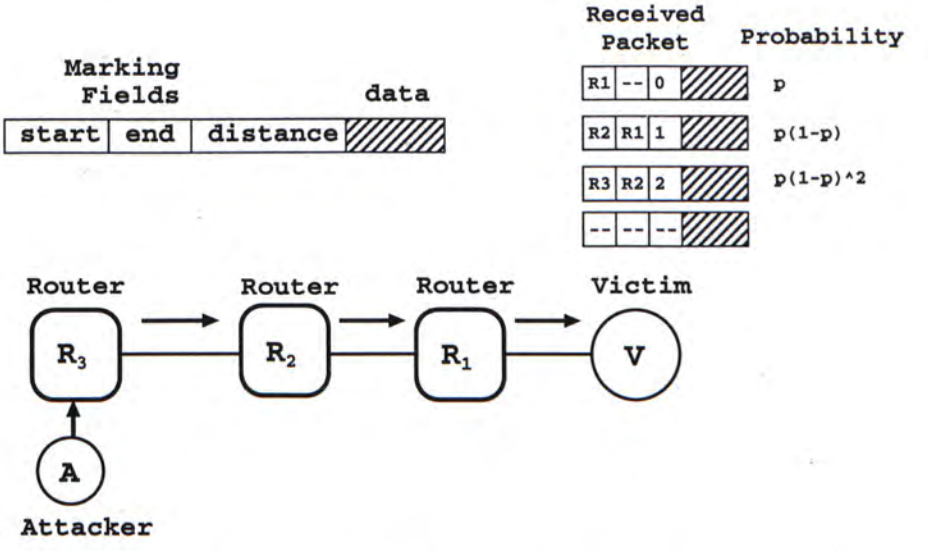

Figure 2.2: Example of a probabilistic edge marking

the *graph construction* algorithm so as to re-construct the attack graph. We present the attack graph construction algorithm in Figure 2.3.

To illustrate the attack graph construction algorithm, let us consider a network which has a tree-like topology depicted in Figure 2.4. In the figure, routers are represented by R_i and the victim site is represented by $\mathcal{V}$. Packets passing through the routers will be marked by the probabilistic edge marking algorithm depicted in Figure 2.1. After some measurement period, the victim site $\mathcal{V}$ will receive packets with the following marking classification:

Distance from $\mathcal{V}$	**Packet Marking Type**
–	un-marked packet
0	(R_1,––,0), (R_5,––,0), (R_8,––,0)
1	(R_2, R_1,1), (R_6, R_5,1), (R_9, R_8,1)
2	(R_3, R_2,2), (R_4, R_2,2), (R_7, R_6,2), (R_{10}, R_9,2)

Algorithm: Attack Graph Construction
Procedure at victim site $\mathcal{V}$

```
Initialize the tree G to have a root node which
      is the victim site V;
filter out unmarked packet;
sort marked packet in ascending order of the value
      of the distance field;
/* attach each marked edge to the tree G */
for (each marked packet w) {
 if (w.distance == 0) {
  insert edge (w.start,V,0) into G;
 }
 else {
  if ((w.end == one of the outermost node in G) and
      (w.distance == that outermost node's distance))
     insert edge (w.start,w.end,w.distance) into G;
 }
}
extract path (R_i..R_j) by enumerating acyclic paths in G;
```

Figure 2.3: Attack graph construction algorithm

Victim site $\mathcal{V}$ uses these marked packets to create an attack graph based on the attack graph construction algorithm depicted in Figure 2.3. From the above example, the attack graph contains four linear paths, they are:

1. **Path 1:** $R_3 \rightarrow R_2 \rightarrow R_1 \rightarrow \mathcal{V}$,
2. **Path 2:** $R_4 \rightarrow R_2 \rightarrow R_1 \rightarrow \mathcal{V}$,
3. **Path 3:** $R_7 \rightarrow R_6 \rightarrow R_5 \rightarrow \mathcal{V}$ and
4. **Path 4:** $R_{10} \rightarrow R_9 \rightarrow R_8 \rightarrow \mathcal{V}$.

2.1.3 Advantages and Disadvantages of Algorithm

It is important to point out that the advantages of the probabilistic marking algorithm are

- One only needs to allocate a small and finite space at the IP packet header so as to store the edge information. As proposed in [SWKA00], this can be achieved using the existing unused space at the IP's packet header.

- The distance field provides an *"ordering relationship"* between edges so that a victim site $\mathcal{V}$ can construct the attack graph from the received marked edges.

- One can use this algorithm to construct the attack graph in real-time (e.g., construction on an attack graph in the midst of a DDoS attack).

One major shortcoming of the probabilistic edge marking algorithm [SWKA00, SP01] is that it does not provide an effective means to *locate* the positions of the potential attackers. In general, the attack graph only provides the topology of traffic targeted to a victim site $\mathcal{V}$. In the following, we present a methodology to identify the potential attackers and at the same time, to quantify the minimum time required to determine the locations of the attackers.

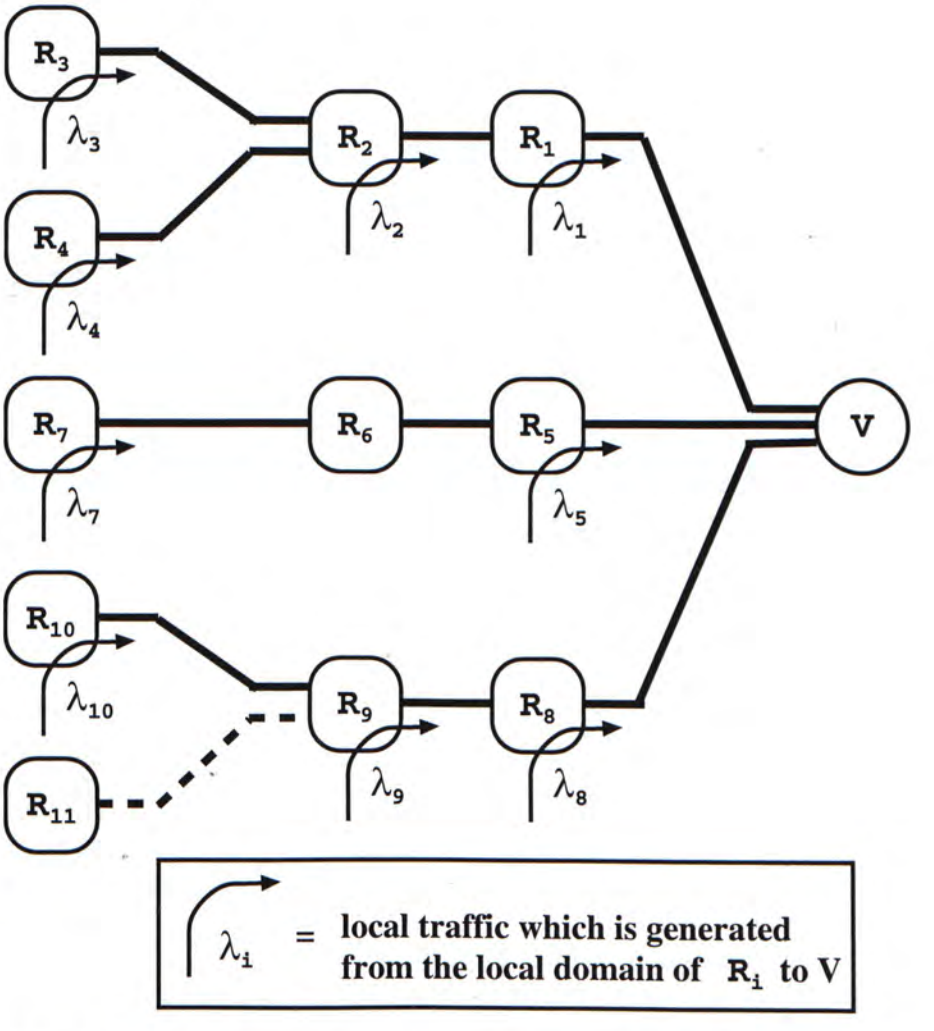

Figure 2.4: Example of attack graph construction: Solid lines represent edges of the constructed attack graph $\mathcal{G}$ while a dotted line represents an edge that is not marked or discovered by the algorithm.

Chapter 3

Attacker Traceback: Linear Topology

We observe that the victim site $\mathcal{V}$ can utilize the marked packets in two ways. First, these marked packets are used to construct an attack graph $\mathcal{G}$. Secondly, one can also use these marked packets to generate *statistical information* so as to traceback the potential attackers. Before we proceed, let us define the following concept.

Definition 1 A "local traffic" of router R_i is the traffic workload generated within the local administrative domain of router R_i.

To illustrate, consider a LAN using a router R_i as its gateway to the Internet, then all packets generated within this LAN to machines outside this LAN domain have to go through the router R_i and these packets are considered as the local traffic of R_i. In the following, we will show that based on the collected marked packets, one can deduce the intensity of the *local traffic* for every router in the attack graph $\mathcal{G}$. Based on the traffic intensity of the local traffic rate at each router, one can determine the possible locations of the attackers. It is important to point out that one major technical difficulty in estimating the local traffic rate of a router is to determine the *"minimum stable time"* t_{min}, which is the minimum time we need to collect the marked packets

so that the calculated local traffic rates are *correct* and *stable*. In the following two subsections, we illustrate how we can determine the local traffic rates and the minimum stable time t_{min}.

3.1 Determination of Local Traffic Rates

To illustrate the methodology, let us consider the following simple but illustrative example (in later chapter, we will extend the methodology to a general network topology). Figure 3.1 illustrates a linear network topology with d routers R_i, where $1 \leq i \leq d$. Router R_i receives its local traffic (e.g. traffic which is generated from its local network domain) to the victim site $\mathcal{V}$. Let λ_i denotes the average local traffic rate, in unit of packets per second, from the router R_i to the victim site $\mathcal{V}$. Each router also performs packet routing/forwarding function for all upstream traffics which are targeted to the victim site $\mathcal{V}$. For example, router R_3 in Figure 3.1 performs packet routing/forwarding for traffic λ_j for $j = 3, 4, \ldots, d$. The objective is that if one can *estimate* the local traffic rate λ_i for *all* these routers from the received marked packets, then based on the intensities of these traffic rates, one can identify the locations of the attackers. Also note that in estimating the local traffic rate λ_i, one cannot simply rely on inspecting the source IP addresses since they can be easily spoofed by the attackers.

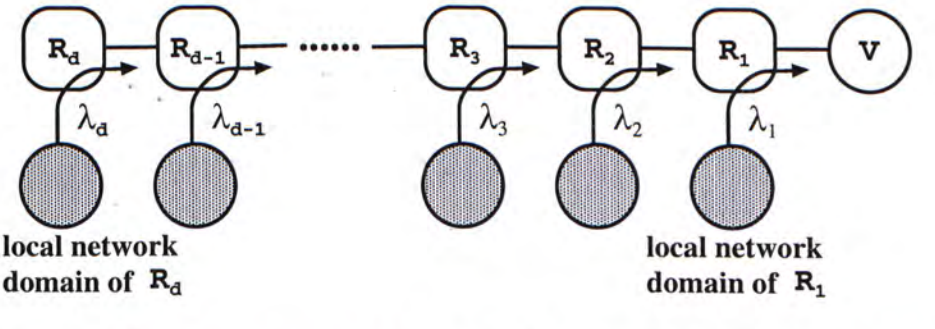

Figure 3.1: Linear topology with local traffic rate λ_i to the victim site $\mathcal{V}$.

Due to the nature of the probabilistic marking algorithm, each router may mark any local or upstream transit packet to the victim site $\mathcal{V}$. If the router R_j decides to mark a transit packet, then in essence, it resets the packet's marking, if there is any, by any upstream routers of R_j. Let $\tilde{N}_j(t)$ be the random variable denoting the number of marked packets received by the victim site $\mathcal{V}$ at time t such that the `start` field is equal to router R_j, for $1 \leq j \leq d$. To compare these random variables $\tilde{N}_j(t)$ for $1 \leq j \leq d$, we need the following definition from [Ros96].

Definition 2 Let X_1 and X_2 be two random variables. We say that X_1 is stochastically larger than X_2, denote as $X_1 \geq_{st} X_2$, if

$$Prob[X_1 > x] \geq Prob[X_2 > x] \quad \forall x.$$

It is clear that if we collect marked packets for a sufficiently long period, then we have the following relationship:

$$\tilde{N}_j(t) \ \geq_{st} \ \tilde{N}_{j+1}(t) \qquad \text{for } j = 1, \ldots, d-1. \tag{3.1}$$

The above relationship holds because

1. Each router carries its local traffic as well as any upstream transit traffic to the victim site $\mathcal{V}$. In a long run, the number of *marked* packets by R_j should be greater than the number of *marked* packets by R_{j+1}.

2. Each router R_j will probabilistically mark a transit packet to the victim site $\mathcal{V}$. Therefore, the router R_j erases any edge marking by an upstream router.

Let $N_j(t)$ be the number of marked packets received by the victim site $\mathcal{V}$ at time t such that the `start` field is equal to router R_j. We have the following relationship:

$$N_j(t) \ = \ \left(\sum_{i=j}^{d} \lambda_i t \right) p(1-p)^{j-1} \qquad j = 1, \ldots, d. \tag{3.2}$$

The term in the parenthesis corresponds to the total number of packets targeted to victim site $\mathcal{V}$ forwarding by router R_j at time t, the remaining term is the probability that these packets are marked by R_j and that these packets are not marked by any downstream routers of R_j. Equation (3.2) is a system of triangular equations, we can re-arrange the above equations and obtain the local traffic rate λ_i at each router R_i as:

$$\lambda_i = \begin{cases} \frac{N_i(t)}{tp(1-p)^{i-1}} - \frac{N_{i+1}(t)}{tp(1-p)^i} & 1 \leq i \leq d-1, \\ \frac{N_d(t)}{tp(1-p)^{d-1}} & i = d. \end{cases} \tag{3.3}$$

3.2 Determination of Minimum Stable Time t_{min}

The remaining technical issue is, to have an accurate estimation of these local traffic rates, we have to make sure that the conditions in Equation (3.1) are satisfied. It is not difficult to observe that these conditions are not satisfied, for example, when the duration of collecting these marked packet is very small (e.g. $t \approx 0$). In the following, we provide an analytical method to derive the minimum stable time t_{min} such that the conditions of Equation (3.1) are satisfied. Once these conditions are satisfied, we can then use the estimation of the local traffic rates based on Equation (3.3) to traceback the potential attackers.

To simplify the notation, let us define:

$$N_j(t) = \lambda'_j t \text{ where } \lambda'_j = \left(\sum_{i=j}^{d} \lambda_i \right) p(1-p)^{j-1} \tag{3.4}$$

Assuming that the random variables $\tilde{N}_j$ and $\tilde{N}_{j+1}$ are Poisson random variables[1] , we can re-formulate the problem of finding the minimum stable time t_{min} as:

$$\text{Prob}[\tilde{N}_j(t_{min}) \geq \tilde{N}_{j+1}(t_{min})] \geq p_{threshold} \quad \forall j \in \{1, \ldots, d-1\}. \tag{3.5}$$

where $p_{threshold}$ is a large probability (e.g., $p_{threshold} = 95\%$). In other words, we want to find the minimum time t_{min} such that, with a very high probability, the number of collected packets marked by router R_j is higher than the number of collected packets marked by router R_{j+1} for $1 \leq j \leq d-1$. Since the random variables $\tilde{N}_j(t)$ and $\tilde{N}_{j+1}(t)$ are Poisson random variables, we have

$$\begin{aligned}
&\text{Prob}[\tilde{N}_j(t_{min}) \geq \tilde{N}_{j+1}(t_{min})] \quad = \\
&\sum_{k=0}^{\infty} \text{Prob}[\tilde{N}_j(t_{min}) \geq k]\text{Prob}[\tilde{N}_{j+1}(t_{min}) = k] = \\
&\sum_{k=0}^{\infty} \left[\sum_{n=k}^{\infty} \frac{(\lambda'_j t_{min})^n}{n!} e^{(-\lambda'_j t_{min})} \right] \frac{(\lambda'_{j+1} t_{min})^k}{k!} e^{(-\lambda'_{j+1} t_{min})} \\
&\qquad \forall j \in \{1, \ldots, d-1\}.
\end{aligned} \tag{3.6}$$

Based on the above expression, one can easily determine the minimum stable time t_{min} using some standard numerical methods [BF88]. Figure 3.2 illustrates the *"local traffic estimation procedure"* to estimate the local traffic rate of each router in an attack graph $\mathcal{G}$.

3.3 Elimination of Attackers

In the previous subsection, we have presented the methodology on how one can estimate the local traffic rate for each router in the attack graph $\mathcal{G}$. In this subsection, we present the algorithm to find the potential attackers and eliminate their attack traffics.

[1] In later chapter, we will illustrate that even if the arrival process is *non-Poisson*, the estimate is still very robust to evaluate t_{min} and the intensity of the local traffic.

Algorithm: Local Traffic Estimation
Procedure at victim site $\mathcal{V}$:

let $\mathcal{P}$ be a linear path to the victim site $\mathcal{V}$;
let $\mathcal{S}_{\mathcal{P}}$ be the set of routers along the path $\mathcal{P}$;
/* Note that $\mathcal{P}$ and $\mathcal{S}_{\mathcal{P}}$ are derived from */
/* the probabilistic edge marking algorithm.*/
set stable = false;
while (stable == false){
/* has not reached t_{min} yet */
collect marked packets for some time;
for (each router R_i in $\mathcal{S}_{\mathcal{P}}$) {
calculate the local traffic λ_i of R_i based on Eq. (3.3);
}
determine whether we have reached the minimum
stable time t_{min} or not for each router in $\mathcal{S}_{\mathcal{P}}$
based on Equation (3.6);
if (t_{min} is reached)
stable = true;
}
output: local traffic rate λ_i for router R_i in $\mathcal{S}_{\mathcal{P}}$.

Figure 3.2: Algorithm to determine local traffic rate for every router in the attack graph.

Given a linear path $\mathcal{P}$ in the attack graph, the victim site $\mathcal{V}$ can choose to reduce its traffic loading by a pre-determined fraction $\mathcal{T}_{cutoff}(\mathcal{P})$. To reduce the traffic, the victim site $\mathcal{V}$ determines the local traffic rates for all routers in the path $\mathcal{P}$, then it sorts these traffic rates in non-increasing order. The victim site $\mathcal{V}$ then sends signal to a subset of routers along the path $\mathcal{P}$ and instructs these routers to stop forwarding their respective local traffics to the victim site $\mathcal{V}$. Note that a router can refuse packet forwarding by examining not the source IP address (which can be spoofed), but rather, by examining data link layer information of a packet, for example, the link address to the attached LAN. Figure 3.3 illustrates the procedure to eliminate the attacker's traffic to victim site $\mathcal{V}$.

Algorithm: Eliminate Potential Attackers along path $\mathcal{P}$

Let $\mathcal{P}$ be a linear path to the victim site $\mathcal{V}$;
Let $\mathcal{S}_{\mathcal{P}}$ be the set of routers along the path $\mathcal{P}$;
Derive the local traffic rate λ_i for router $R_i \in \mathcal{S}_{\mathcal{P}}$ based on the "local traffic estimation procedure";
Sort $\{\lambda_i\}$ in non-increasing order and let the sorted sequence be $\{\acute{\lambda}_i\}$;
Find the *"minimum"* i^* such that $\frac{\sum_{j=1}^{i^*} \acute{\lambda}_j}{\sum_{k=1}^{d} \lambda_k} \geq \mathcal{T}_{cutoff}(\mathcal{P})$;
Send control signals to routers $\{R_j\}_{1 \leq j \leq i^*}$ so that routers can stop their local traffics to victim site $\mathcal{V}$;

Figure 3.3: Algorithm to find potential attackers and eliminate their attack traffics to $\mathcal{V}$.

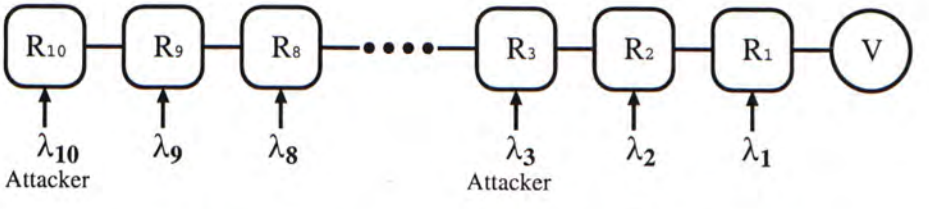

Figure 3.4: Example to estimate the local traffic rates

To illustrate, let us consider an example in Figure 3.4. We consider a linear topology with 10 routers. Routers R_1, R_2, R_4 to R_9 carry normal local traffic with a rate of 10 pkts/sec to the victim site $\mathcal{V}$. There are two attackers in the network and they are located at routers R_3 and R_{10}. The attacker in R_3 generates an attack traffic rate of 80 pkts/sec to $\mathcal{V}$ while the attacker in R_{10} generates an attack traffic rate of 100 pkts/sec to $\mathcal{V}$. The marking probability p for the Edge Marking Algorithm is set to 0.1 and the stability threshold $p_{threshold}$ is set to 95%. In this example, the victim site $\mathcal{V}$ calculates the percentage of stability for every 10 seconds. Table 3.1 illustrates that after going through the loop of the traffic estimation procedure six times, we can

accurately estimate the local traffic rate at each router R_i for $i = 1, \ldots, 10$. In other words, it only takes around 60 seconds to find the potential attackers in R_3 and R_{10}. Table 3.2 illustrates the percentage of the local traffic at each

Time (sec)	**Stability** (%)	R_3 **Traffic** (pkts/sec)	R_{10} **Traffic** (pkts/sec)	Other R_i **Average Traffic** (pkts/sec)
10.0	43	99.18	92.92	7.24
20.0	43	74.97	85.18	12.04
30.0	67	72.43	88.62	12.20
40.0	76	66.19	100.67	11.21
50.0	92	78.90	103.25	9.38
60.0	97	78.58	100.24	10.04

Table 3.1: Estimation of the local traffic rates in R_1 to R_{10}.

Router	**% of local traffic to total traffic**	**Router**	**% of local traffic to total traffic**
R_1	3%	R_6	5%
R_2	4%	R_7	4%
R_3	30%	R_8	5%
R_4	5%	R_9	3%
R_5	3%	R_{10}	38%

Table 3.2: Percentage of local traffic to the total received traffic by $\mathcal{V}$ after 60.0 seconds

router at the end of 60 seconds, which is the time that the estimation of the local traffic rates becomes stable. As we can observe from the table, the victim site $\mathcal{V}$ can accurately estimate the intensity of the local traffic of each router in the attack graph. If the victim site $\mathcal{V}$ wants to reduce the total traffic by 50%, one can set the parameter $\mathcal{T}_{cutoff}(\mathcal{P})$ to 50%. The victim site $\mathcal{V}$ can then send control signals to routers R_{10} and R_3 and local traffic from these two routers will be eliminated according to the procedure in Figure 3.3.

Chapter 4

Attacker Traceback: General Topology

In the previous chapter, we have presented the methodology of how one can deduce the *local traffic rate* of each router as well as the *minimum stable time* t_{min} such that we can determine the locations of the potential attackers in a linear network topology. In this chapter, we extend the methodology to a general attack graph topology. The basic ideas are similar to the previous chapter but we need to refine our conditions under the general topology setting.

4.1 Determination of Local Traffic Rates

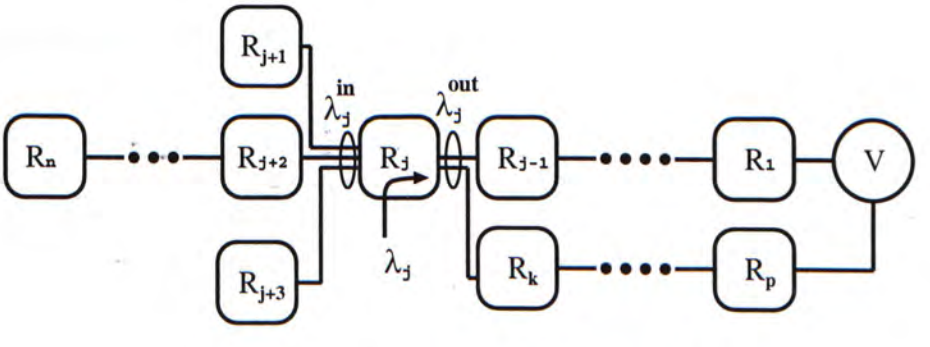

Figure 4.1: A general attack graph $\mathcal{G}$

Figure 4.1 illustrates a general topology wherein each router may be connected by several upstream and downstream routers. For example, router R_j in the figure has three upstream routers R_{j+1}, R_{j+2} and R_{j+3} and two downstream routers R_{j-1} and R_k. For this attack graph, the furthest router from the victim site $\mathcal{V}$ is router R_n, which has a distance of $d \geq 1$ hops away from $\mathcal{V}$. Each router receives its local traffic from its network domain, some of these traffics are destined to the victim site $\mathcal{V}$. Let λ_j denotes the average local traffic rate, in unit of packets per second, from the router R_j to the victim site $\mathcal{V}$. Let λ_j^{in} denotes the total average traffic rate from all upstream routers of R_j to the victim site $\mathcal{V}$ and λ_j^{out} denotes the total average traffic rate from router R_j to all its downstream routers which are targeted to the victim site $\mathcal{V}$. Again, the goal is to deduce the *local traffic rate* λ_j for all routers in an attack graph, then based on their local traffic intensities, we can identify the locations of the potential attackers.

Let $\mathcal{G}$ denotes the attack graph based on the attack graph construction method we described in Section 2. We say that a router is a *leaf* router in the attack graph $\mathcal{G}$ if it is not connected to any upstream router. For example, in Figure 4.1, router R_n is a leaf router. All other routers are called *internal* routers. Let (i, j) denotes an edge between router R_i and R_j, we define $\lambda_{(i \to j)}$ as the average traffic rate to the victim site $\mathcal{V}$ that passes through the edge (i, j) between router R_i and R_j.

The average local traffic rate λ_j for router $R_j \in \mathcal{G}$ is:

$$\lambda_j = \begin{cases} \lambda_j^{out} & \text{if } R_j \text{ is a leaf router,} \\ \lambda_j^{out} - \lambda_j^{in} & \text{if } R_j \text{ is an internal router.} \end{cases} \tag{4.1}$$

where λ_j^{in} and λ_j^{out} are the average traffic rates into and out of router R_j

respectively. These average traffic rates can be computed by

$$\lambda_j^{in} = \sum_{\substack{\forall (i,j) \text{ where } R_i \text{ is an} \\ \text{upstream router of } R_j}} \lambda_{(i \to j)}, \tag{4.2}$$

$$\lambda_j^{out} = \sum_{\substack{\forall (j,k) \text{where } R_k \text{ is a} \\ \text{downstream router of } R_j}} \lambda_{(j \to k)}. \tag{4.3}$$

Therefore, if we can estimate the average traffic rate $\lambda_{(i \to j)}$ for all marked edges (i, j) in the attack graph $\mathcal{G}$, then we can deduce the average local traffic rate of each router based on Equations (4.1) to (4.3). In Figure 4.2, we present the procedure to estimate $\lambda_{(i \to j)}$ for each marked edge (i, j) in $\mathcal{G}$.

Let $\tilde{N}_j^{out}(t)$ be the random variable denoting the number of marked packets received by the victim site $\mathcal{V}$ at time t such that the `start` field is equal to router R_j. Let $\tilde{N}_j^{in}(t)$ be the random variable denoting the number of marked packets received by the victim site $\mathcal{V}$ at time t such that the `end` field is equal to router R_j. After a sufficient amount of time in collecting these marked packets, we have the following relationship:

$$\tilde{N}_j^{out}(t) \geq_{st} \tilde{N}_j^{in}(t) \qquad \forall R_j \in \mathcal{G}. \tag{4.4}$$

The above relationship holds because

1. There is non-negative local traffic originated from router R_j to the victim site $\mathcal{V}$, therefore, the number of *marked* output packets from R_j can be greater than the number of *marked* input packets to R_j in the long run.

2. The probabilistic edge marking algorithm will mark any transit packet to $\mathcal{V}$ from the upstream of router R_j. Therefore, the router R_j may erase any edge marking of a transit packet from its upstream routers.

4.2 Determination of Minimum Stable Time t_{min}

The remaining issue is that to have an accurate estimation of the local traffic rate λ_j, we have to guarantee that the conditions in Equation (4.4) are satisfied. Once the conditions of Equation (4.4) are satisfied, we can then estimate local traffic rate λ_j based on Equation (4.1) - (4.3) to traceback the potential attackers. In the following, we provide an analytical method to derive the minimum stable time t_{min} such that the conditions of Equation (4.4) are satisfied.

Algorithm: Estimating the Average Traffic Rate $\lambda_{(i\to j)}$ for each edge (i,j) in $\mathcal{G}$

Let $\mathcal{G}$ be an attack graph with root $\mathcal{V}$;
Each marked edge (i,j) in $\mathcal{G}$ has $(R_i, R_j, d_{(i\to j)}, N_{(i\to j)})$
/* R_i = `start address of the marked edge` (i,j)
R_j = `end address of the marked edge` (i,j)
$d_{(i\to j)}$ = `hop count between` R_i `and victim` $\mathcal{V}$
$N_{(i\to j)}$ = `the number of times this marked edge` (i,j) `has been collected` */
Set $\lambda_{(i\to j)} = 0$ for all edges (i,j) in $\mathcal{G}$;
For each edge in $\mathcal{G}$ {
`/* go through each edge in` $\mathcal{G}$ `*/`
add $\lambda_{(i\to j)}$ by $\frac{N_{(i\to j)}}{tp(1-p)^{d_{(i\to j)}}}$
}
output: average traffic rate $\lambda_{(i\to j)}$ for each edge (i,j) in $\mathcal{G}$.

Figure 4.2: Procedure to estimate $\lambda_{(i\to j)}$ for every marked edge (i,j) in $\mathcal{G}$.

Let $N_j^{out}(t)$ be the number of marked packets received by the victim site $\mathcal{V}$ at time t such that the `start` field is equal to router R_j. Let $N_j^{in}(t)$ be the number of marked packets received by the victim site $\mathcal{V}$ at time t such that

the end field is equal to router R_j. We have the following relationship:

$$N_j^{in}(t) = \sum_{\forall(i,j)} \lambda_{(i\to j)}(t) p(1-p)^{d_{(i\to j)}-1} t, \tag{4.5}$$

$$N_j^{out}(t) = \sum_{\forall(j,k)} \lambda_{(j\to k)}(t) p(1-p)^{d_{(j\to k)}-1} t. \tag{4.6}$$

where $\lambda_{(i\to j)}(t)$ and $\lambda_{(j\to k)}(t)$ are the traffic rate estimation of the corresponding edge (i,j) and (j,k) at time t according to the procedure given in Figure 4.2, $d_{(i\to j)}$ and $d_{(j\to k)}$ are the hop count from router R_i to victim $\mathcal{V}$ and from router R_j to victim $\mathcal{V}$ respectively.

To simplify notation, let us define $\lambda'^{in}_j(t)$ and $\lambda'^{out}_j(t)$ as:

$$\lambda'^{in}_j(t) = \sum_{\forall(i,j)} \lambda_{(i\to j)}(t) p(1-p)^{d_{(i\to j)}-1},$$

$$\lambda'^{out}_j(t) = \sum_{\forall(j,k)} \lambda_{(j\to k)}(t) p(1-p)^{d_{(j\to k)}-1},$$

$$N_j^{in}(t) = \lambda'^{in}_j(t)\, t \quad \text{and} \quad N_j^{out}(t) = \lambda'^{out}_j(t)\, t \tag{4.7}$$

We can re-formulate the problem of finding the minimum stable time t_{min} such that:

$$\text{Prob}[\tilde{N}_j^{out}(t_{min}) \geq \tilde{N}_j^{in}(t_{min})] \geq p_{threshold}\ \forall R_j \in \mathcal{G}. \tag{4.8}$$

where $p_{threshold}$ is a large probability (e.g., $p_{threshold} = 95\%$). The formulation implies to find the minimum time such that with a very high probability, the number of collected packets marked by the router R_j is higher than the number of collected packets marked by the upstream routers of R_j, for all routers in the attack graph $\mathcal{G}$. Assuming that the random variable is a Poisson process[1]

[1] Again, in later chapter, we will illustrate that the estimate is still very robust to evaluate t_{min} even if the arrival process is non-Poisson.

, we have

$$\begin{aligned}
&\text{Prob}[\tilde{N}_j^{out}(t_{min}) \geq \tilde{N}_j^{in}(t_{min})] \\
&= \sum_{k=0}^{\infty} \text{Prob}[\tilde{N}_j^{out}(t_{min}) \geq k]\text{Prob}[\tilde{N}_j^{in}(t_{min}) = k] \\
&= \sum_{k=0}^{\infty} \left[\sum_{n=k}^{\infty} \frac{\left[\lambda'^{out}_j(t_{min})t_{min}\right]^n}{n!} e^{-\left[\lambda'^{out}_j(t_{min})t_{min}\right]} \right] \times \\
&\qquad \frac{\left[\lambda'^{in}_j(t_{min})t_{min}\right]^k}{k!} e^{-\left[\lambda'^{in}_j(t_{min})t_{min}\right]} \\
&\qquad\qquad \forall R_j \in \mathcal{G}. \qquad (4.9)
\end{aligned}$$

Again, we can easily determine the minimum stable time t_{min} using some standard numerical methods [BF88].

Chapter 5

Simulations

In this chapter, we perform simulations to illustrate the effectiveness in locating the attackers. In particular, we show the correctness and robustness of using Equation (3.6) and (4.9) to find the minimum stable time t_{min} such that we can compute the intensities of the local traffic rates for all routers in the attack graph $\mathcal{G}$. Note that a smaller value of t_{min} implies that we can find the locations of the attackers earlier. We illustrate various factors which can influence the values of the minimum stable time t_{min}. We also extend the analysis of the attacker location scheme to a general network topology. In the first two sets of simulations, we use a linear network topology as depicted in Figure 5.1. To derive the attack graph, the victim $\mathcal{V}$ informs the participating routers to mark packets with probability $p = 1/25$. To estimate whether we have reached the stability conditions specified by Equation (3.6) or (4.9), we set $p_{threshold} = 95\%$. In other words, the estimation of the local traffic rates is highly accurate. We define a variable, **Attack Traffic Ratio** (ATR), which is the ratio of the attacker's average traffic rate to the normal average local traffic. For example, if the normal average local traffic is 100 pkts/sec and ATR is set to 20, then the attacker's average traffic rate is equal to 2000 pkts/sec.

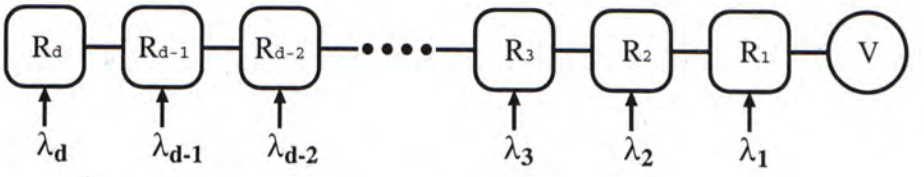

Figure 5.1: Network topology for Simulation 1 and 2.

5.1 Simulation 1 - Correctness and robustness of estimating the minimum stable time t_{min}

This set of simulations evaluates the correctness of the estimation of the minimum stable time t_{min} based on the mathematical model (3.6) presented in Section 3.1. We show that the estimation is robust and accurate for different traffic arrival processes, for example, Poisson arrival process, constant arrival rate or burst mode and even packet generated under the Markov-Modulated Poisson Process (MMPP) [Nel95]. We demonstrate the performance tradeoff with different values of the parameter $p_{threshold}$. For this set of simulations, we maintain the normal average local traffic to 100 pkts/sec. The ATR is set to 20 (in other words, the attacker's average traffic rate is 2000 pkts/sec).

5.1.1 Simulation 1.A - Influence on t_{min} by different packet arrival processes

In this simulation, we use a linear topology in Figure 5.1 with 25 routers (e.g., $d = 25$). There is one attacker and he is located at the furthest router R_{25}. The normal traffic and the attack traffic are generated using three methods: (1) Poisson process, (2) constant rate (e.g., an average rate of 100 pkts/sec implies that every 0.01 second, there is a new packet generated by a router), (3) burst rate (e.g., an average rate of 100 pkts/sec implies that we generate 100 pkts in one burst for every second). To compute the minimum stable time t_{min}, we

use the mathematical derivation in Equation (3.6). Table 5.1 illustrates that with different values of t_{min}, our theoretical computed $p_{threshold}$ value is very close to the simulated $p_{threshold}$ values for different traffic generation processes. This indicates that our methodology is very accurate and robust in estimating the minimum stable time t_{min} and, regardless of the packet arrival processes, it takes around 25 seconds to determine the local traffic rates of all routers in the attack graph $\mathcal{G}$.

t_{min} (sec)	$p_{threshold}$ (%)			
	Theoretical Computed Values	**Simulation**		
		Poisson	**Constant**	**Burst**
5.0	79.46	79.37	79.69	79.88
10.0	87.35	87.36	87.36	87.32
15.0	91.80	91.74	91.78	91.84
20.0	94.55	93.93	94.48	94.59
25.0	96.31	96.19	96.21	96.54

Table 5.1: Minimum Stable Time: Theoretical vs. Simulation Results for different packet arrival processes

5.1.2 Simulation 1.B - Influence on t_{min} by different packet arrival processes under MMPP

We also consider other ways to generate normal and attack traffic. In particular, we consider a two-state Markov-Modulated Poisson Process $\{X(t)\}$ where $X(t) \in \{0, 1\}$. Let λ (μ) be the transition rate from state 0 (state 1) to state 1 (state 0). If the state of this Markov process is in state 1, then packet generation is enabled. If the state of the process is in state 0, then no packet is generated into the system. Again, the packet generation at state 1 can be classified into three types, namely, (1) Poisson process with an average rate of

100 pkts/sec, (2) constant rate (e.g., an average rate of 100 pkts/sec implies that every 0.01 second, there is a new packet generated by a router), (3) burst rate (e.g., an average rate of 100 pkts/sec implies that we generate 100 pkts in one burst for every second). We consider two different sets of MMPP, (a) $\lambda = 2.0, \mu = 1.0$ and, (b) $\lambda = 10, \mu = 30$. Table 5.2 and 5.3 depict the influence on the minimum stable time t_{min} under these two MMPP settings. Again, we observe that it is very accurate and robust in estimating the minimum stable time t_{min} even when the packet arrival process is non-Poisson. Another important observation is that we can quickly estimate t_{min}, in particular, we need around 25 seconds to reach the 95% stability condition. Therefore, this implies we can quickly determine the locations of the attackers.

t_{min} (sec)	$p_{threshold}$ (%)			
	Theoretical Computed Values	**MMPP Pkt Generation** $\lambda = 2$, $\mu = 1$		
		Poisson	**Constant**	**Burst**
5.0	79.46	79.70	79.70	79.50
10.0	87.35	88.40	87.90	87.90
15.0	91.80	91.50	92.00	90.80
20.0	94.55	94.40	94.90	94.20
25.0	96.31	95.50	96.20	96.60

Table 5.2: Minimum Stable Time: Theoretical vs. Simulation Results for different packet arrival processes under MMPP with $\lambda = 2.0$ and $\mu = 1.0$

5.1.3 Simulation 1.C - Influence on t_{min} and variance of traffic rate estimation by different $p_{threshold}$

Equation (3.6) indicates that if the stochastic relationship in Equation (3.1) are satisfied with a high probability $p_{threshold}$, the estimation of local traffic will then be accurate. The accuracy can be expressed by the variance of the

t_{min} (sec)	$p_{threshold}$ (%)			
	Theoretical Computed Values	**MMPP Pkt Generation** $\lambda = 10$, $\mu = 30$		
		Poisson	**Constant**	**Burst**
5.0	79.46	81.50	77.50	78.30
10.0	87.35	88.00	86.80	86.20
15.0	91.80	91.70	92.70	90.50
20.0	94.55	93.40	93.90	94.60
25.0	96.31	97.30	95.90	96.60

Table 5.3: Minimum Stable Time: Theoretical vs. Simulation Results for different packet arrival processes under MMPP with $\lambda = 10$ and $\mu = 30$.

local traffic rate estimation. In this simulation, we illustrate how the parameter $p_{threshold}$ affects the minimum stable time t_{min} and the quality of the attacker's traffic rate estimation. Figure 5.2 illustrates that for different values of $p_{threshold}$, we have different values of t_{min} and its associated variance of attacker's traffic rate. Figure 5.2 shows that the variance in the estimation of the attacker's traffic rate approaches to zero (e.g., which represents a highly accurate estimation of attacker's traffic) when we set $p_{threshold} \geq 80\%$. Of course, a higher value of $p_{threshold}$ also implies a larger value of t_{min}, which can affect how quickly we can determine the locations of the attackers. Nevertheless, because t_{min} is less than 60 seconds, this implies that we can quickly response to the DDoS attack and locate the attackers.

5.2 Simulation 2 - Factors which influence the minimum stable time t_{min}

In this set of simulations, we illustrate how the length of an attack path, the number of attackers, and the attack traffic ratio ATR can affect the values of the minimum stable time t_{min}. For this set of simulations, we maintain the

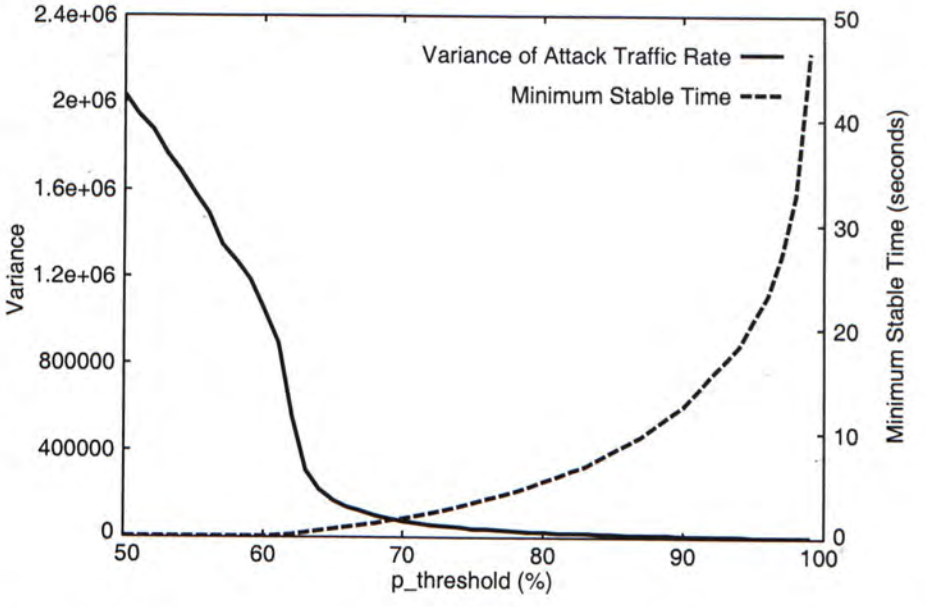

Figure 5.2: Influence on t_{min} and variance of the attacker's traffic rate estimation by different $p_{threshold}$.

normal average local traffic to 100 pkts/sec. The ATR is set to 20 (or the attacker's average traffic rate is 2000 pkts/sec).

5.2.1 Simulation 2.A - Influence on t_{min} by different length of the attack path

In this simulation, the relationship between the length of an attack path and the minimum stable time t_{min} is analyzed. We use a linear network topology with varying length of attack path and compute the minimum stable time t_{min}. We assume there is one attacker and he is located at the *furthest* router (e.g., R_d) Figure 5.3 shows that as the length of the attack path increases, the minimum stable time t_{min} also increases. The reason for this linear growth is that the victim site $\mathcal{V}$ needs to take a longer time to collect sufficient number of *marked* packets from the furthest router.

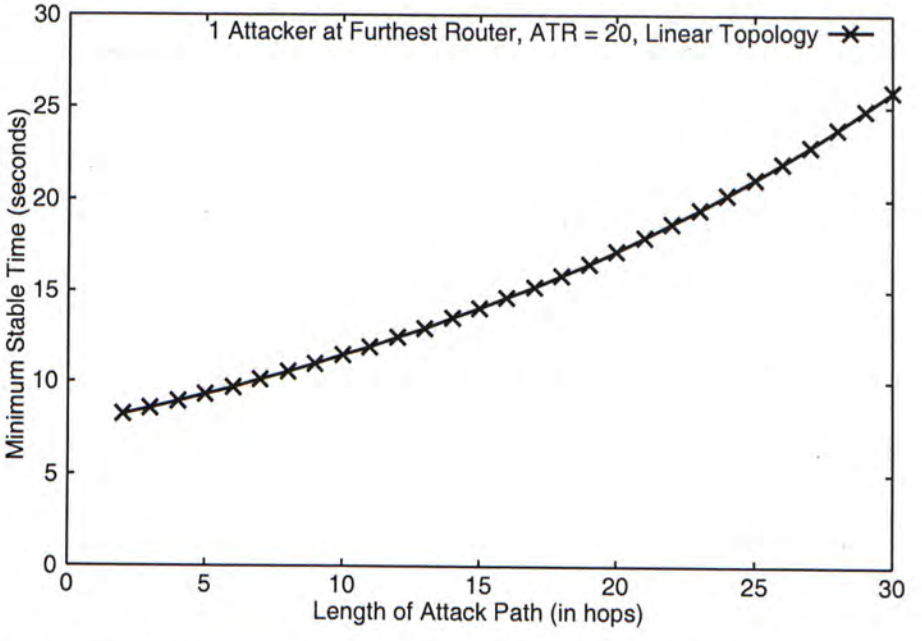

Figure 5.3: Influence on t_{min} by different length of the attack path

5.2.2 Simulation 2.B - Influence on t_{min} by the relative positions of the attackers

In this simulation, we study how the number of attackers and their relative positions influence the value of the minimum stable time t_{min}. We consider three attackers in a linear network topology and these three attackers are distributed by three different configurations. For configuration I, these three attackers are located at routers R_1, R_2 and R_3 (e.g., the 3 closest routers to the victim $\mathcal{V}$). For configuration II, these three attackers are located at the three furthest routers from the victim $\mathcal{V}$. In configuration III, these three attackers are *evenly distributed* along the linear network, for example, they are located at routers $R_{\lfloor \frac{d*i}{3} \rfloor}$ where d is the length of attack path and i=1,2,3. Figure 5.4 illustrates the minimum stable time t_{min} when ATR= 20 and $p_{threshold}$ = 95%. We observe that when the attackers are closer to the victim $\mathcal{V}$ (e.g., configuration I), the achieved minimum stable time t_{min} is lower than other

configurations. The minimum stable time t_{min} achieves highest value when attackers are evenly distributed in the network (e.g., configuration III). The reason for this ordering is that the estimation of local traffic rate at the furthest router takes longer time to become stable while the estimation of local traffic rates at the nearest router takes less time. Regardless of the distribution of the attackers' positions, we can determine their locations within 30 seconds, which shows the effectiveness of the proposed methodology.

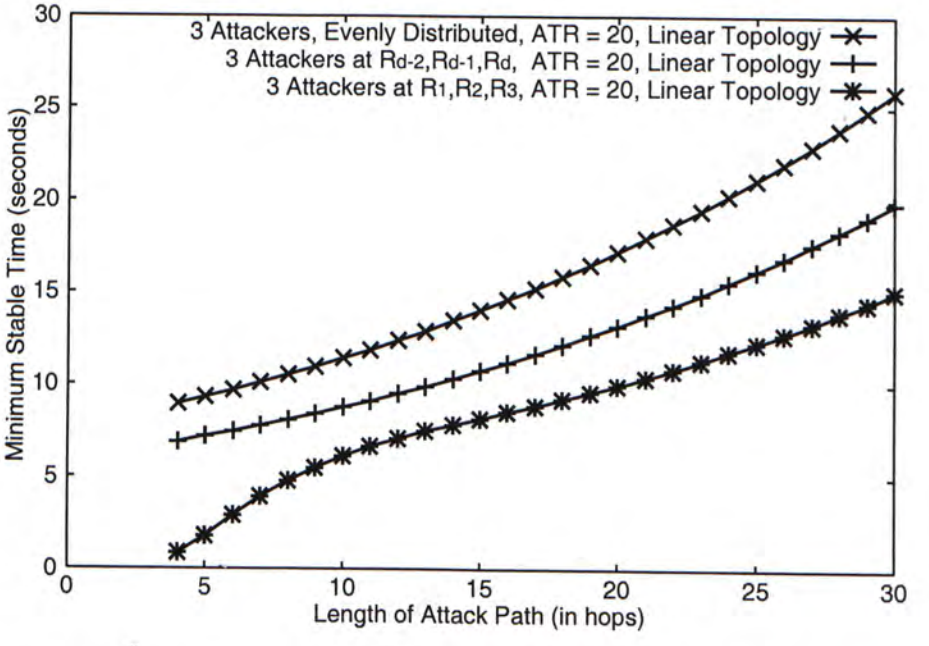

Figure 5.4: Influence on t_{min} by the relative positions of the attackers

5.2.3 Simulation 2.C - Influence on t_{min} by different ATR and different length of the attack path

In this simulation, we illustrate how the attack traffic ratio (ATR) and the length of the attack path affect the value of the minimum stable time t_{min}. We consider a linear topology and we compute the minimum stable time t_{min} with varying attack traffic ratio as well as the length of the attack path. Since

the previous result suggested that t_{min} takes on the largest value when the attacker is located at the furthest router, therefore, we consider the worst scenario that one attacker is located at the furthest router R_d (note that this is equivalent to the worst case analysis). Figure 5.5 illustrates the achieved minimum stable time t_{min} under different values of ATR and the length of the attack path. We observe that if the attack path is longer, the minimum stable time t_{min} has a larger value. On the other hand, when the attack traffic ratio is high (which is usually the case for a DoS attack), the minimum stable time t_{min} has a lower value. In general, the t_{min} is upper bounded by 30 seconds. Again, this shows that we can quickly discover the locations of the potential attackers under a DDoS attack.

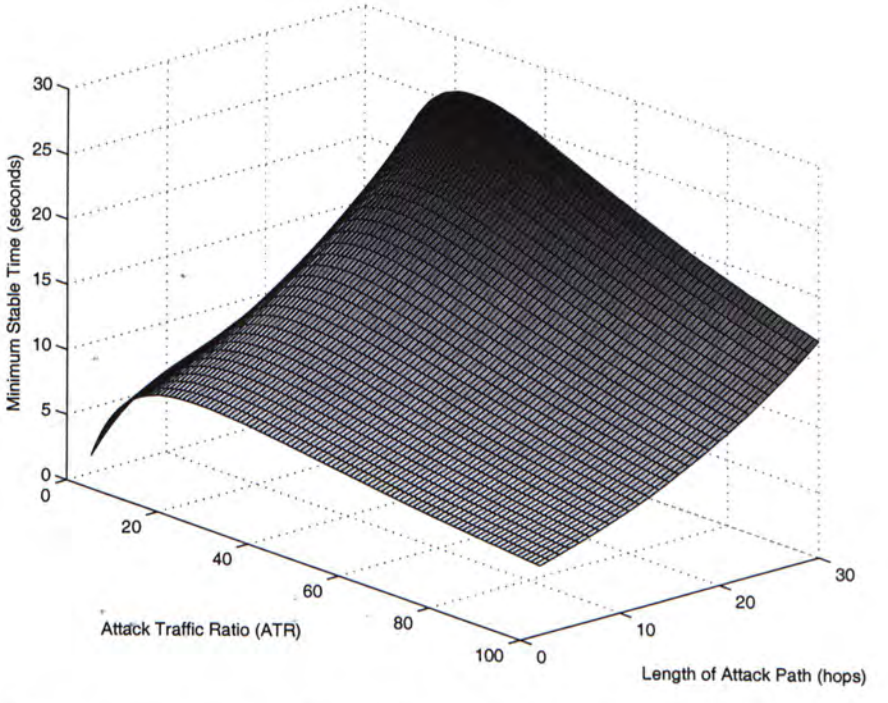

Figure 5.5: Influence on t_{min} by different ATR and different length of the attack path

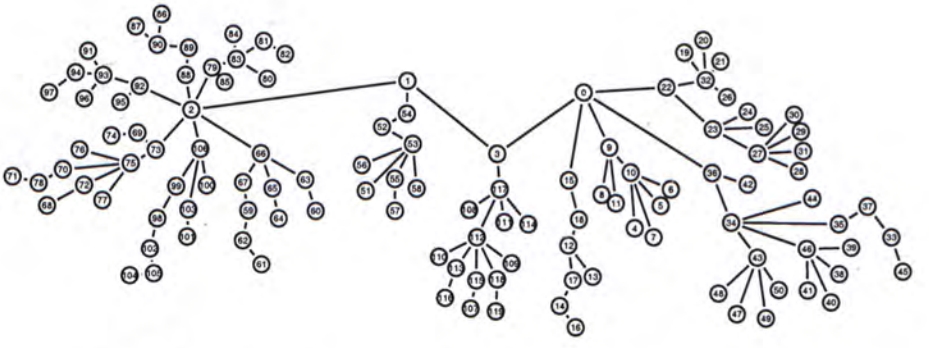

Figure 5.6: General network topology with 120 routers

5.3 Simulation 3 - Extension to General Network Topology

In the previous set of simulations, we analyze the performance of the attacker location scheme by assuming the topology is linear. In this set of simulations, we extend the performance study to the general network topology. We evaluate the minimum stable time t_{min} of the attacker location scheme based on the mathematical model (4.9) presented in Chapter 4. Figure 5.6 shows a general network topology of 120 routers. We assume the victim $\mathcal{V}$ is located at the router R_0 and the attackers are randomly assigned to different locations of disjoint attack paths. We maintain normal average local traffic to 10 pkts/sec and vary the attackers' average traffic rates to 8000, 9000, and 10000 pkts/sec. (or ATR = 800, 900, 1000).

5.3.1 Simulation 3.A - Influence on t_{min} by different ATR and different diameter of the network topology

In this simulation, we vary the diameter of the general attack graph and evaluate the average minimum stable time t_{min} under different attackers' traffic

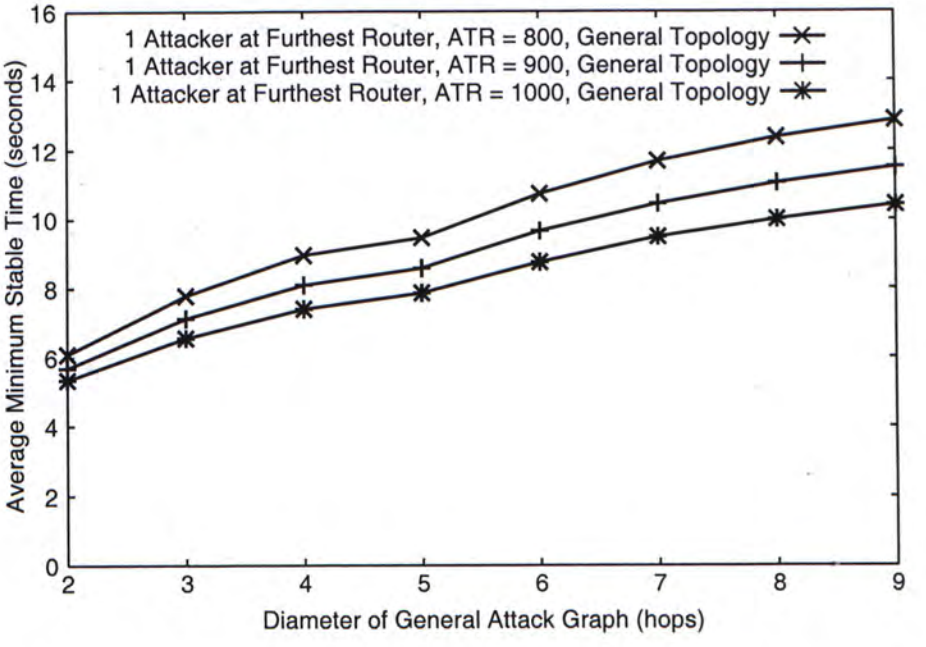

Figure 5.7: Influence on t_{min} by different ATR and different diameter of the network topology

rates. Figure 5.7 illustrates that as the diameter of the general attack graph increases, the average minimum stable time t_{min} also increases. The reason is that it takes a longer time to collect sufficient amount of marked packets from the furthest router. Moreover, when the attack traffic ratio (ATR) is high (which is a common scenario for a denial-of-service attack), the average minimum stable time t_{min} will be smaller. This simulation shows that we can determine the locations of the attackers within 14 seconds in this general network topology, which shows the effectiveness of our proposed methodology.

5.3.2 Simulation 3.B - Influence on t_{min} by different number of attackers

In this simulation, we vary the number of attackers in the general attack graph and evaluate the average minimum stable time t_{min} under different attackers'

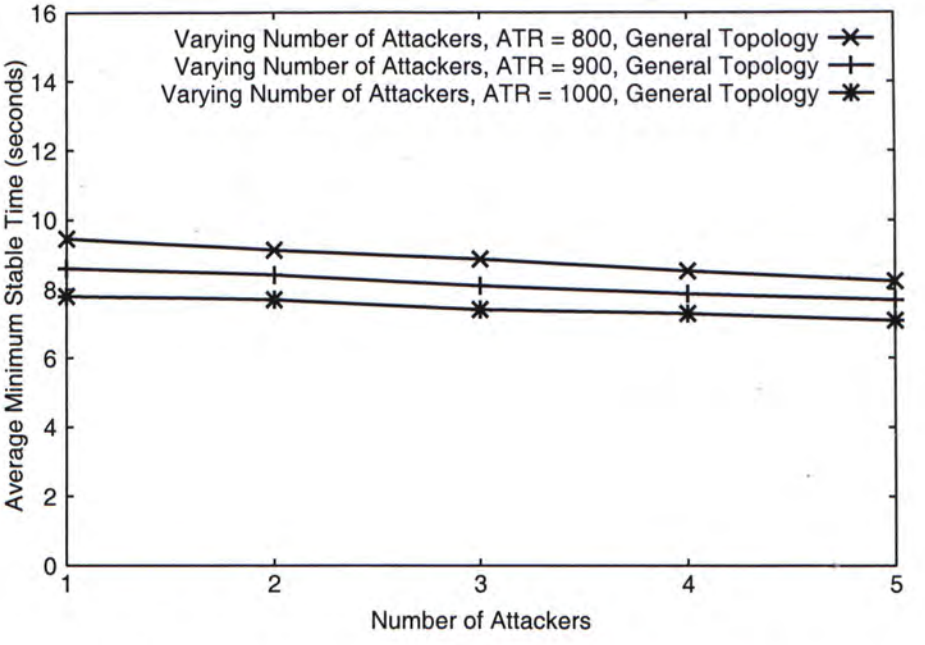

Figure 5.8: Influence on t_{min} by different number of attackers

traffic rates. The attackers are located at the furthest routers of the attack graph. (It is equal to worst case study.) Figure 5.8 shows that as the number of attackers increases, the average minimum stable time t_{min} decreases. The reason for the decrease of t_{min} is that larger number of attackers, more marked packets will be collected , faster to satisfy the stability conditions of the local traffic estimation. This illustrates we can quickly determine the locations of the attackers under a DDoS attack.

5.4 Simulation 4 - Extension to Internet Topology

In the previous set of simulations, we evaluate the performance of our attacker location scheme in a small-sized general network topology. In this set of simulations, we perform a more comprehensive performance study by using

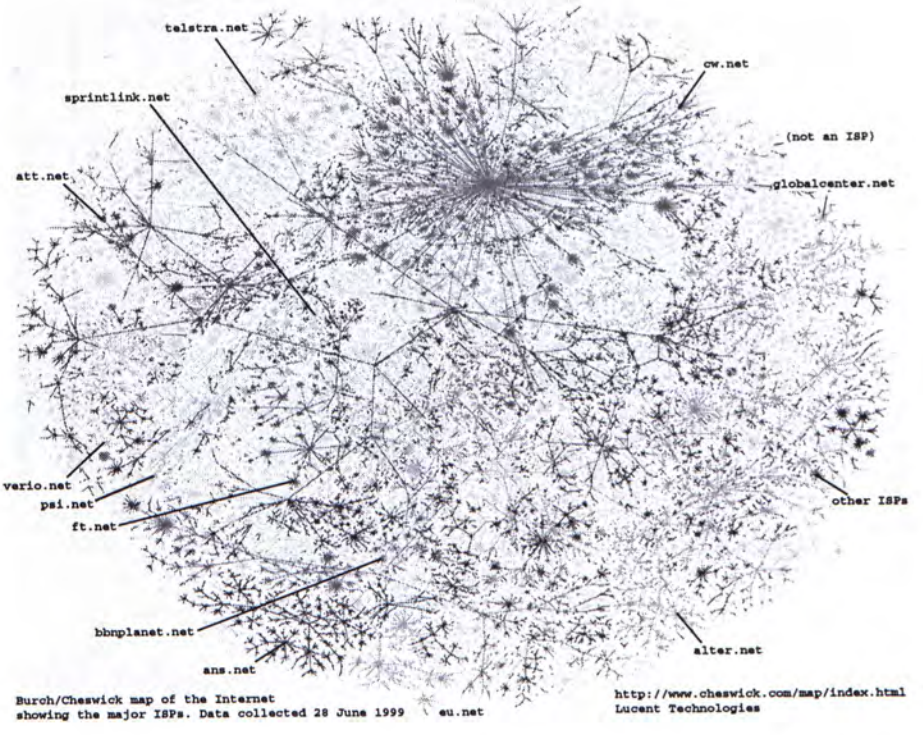

Figure 5.9: Internet map generated by the traceroute dataset

a large-sized Internet topology obtained from Lucent Bell Labs [tra99]. Figure 5.9 shows an Internet map[1] generated by the traceroute dataset. The testing dataset in our simulations contains 10,000 distinct routers and the maximum height (or the maximum hop count from the victim site $\mathcal{V}$ to the furthest router in the graph) is 23. The source of traceroute is considered as the victim site $\mathcal{V}$ and the traceroute dataset is considered as the map of the upstream routers. We use this dataset and repeat similar simulations as in Simulation 3. We maintain normal average local traffic to 1 pkt/sec and the attackers' average traffic rate to 10000 pkts/sec (ATR = 10000).

[1]Image from `http://research.lumeta.com/ches/map` Internet Mapping Project.

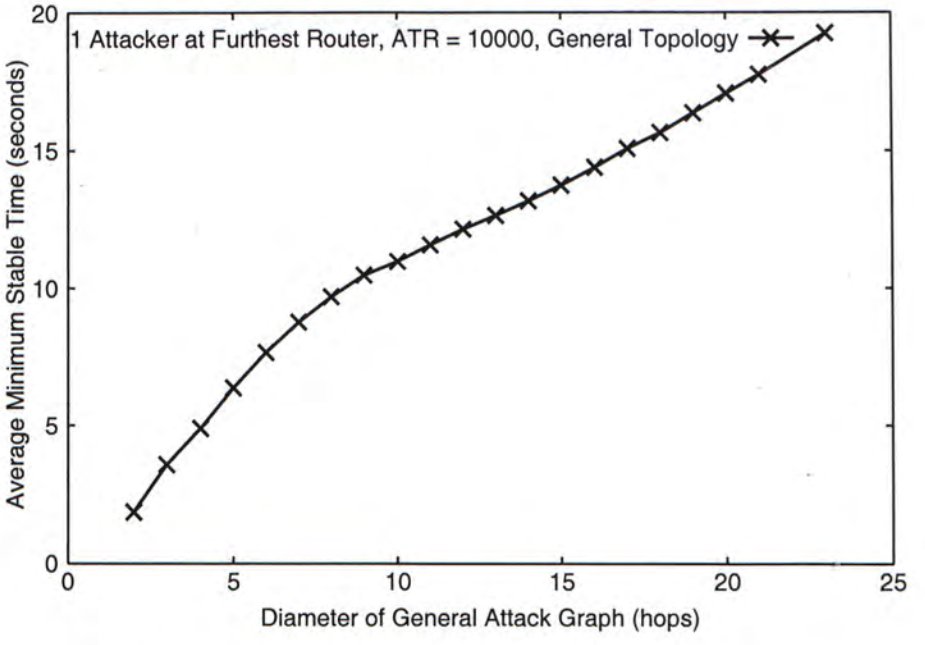

Figure 5.10: Influence on t_{min} by different diameter of the network topology

5.4.1 Simulation 4.A - Influence on t_{min} by different diameter of the network topology

In this simulation, we study the relationship between the diameter of the general attack graph $\mathcal{G}$ and the average minimum stable time t_{min} in Internet topology. The main difference between a small-sized topology and a large-sized topology is that the aggregate transit traffic for routers which are closer to a victim site $\mathcal{V}$ is significantly larger than those routers which are further away from the victim site $\mathcal{V}$ in a large-sized topology. Figure 5.10 illustrates the average minimum stable time t_{min} when there is a single attacker. Again, we observe that when the attacker is closer to the victim site $\mathcal{V}$, then we can determine the location of the attacker earlier. One important point is that even for this large network, it only takes less than 20 seconds to determine the local traffic rate of each router and locate the attacker. This implies that our methodology is very robust and accurate in determining the locations of the

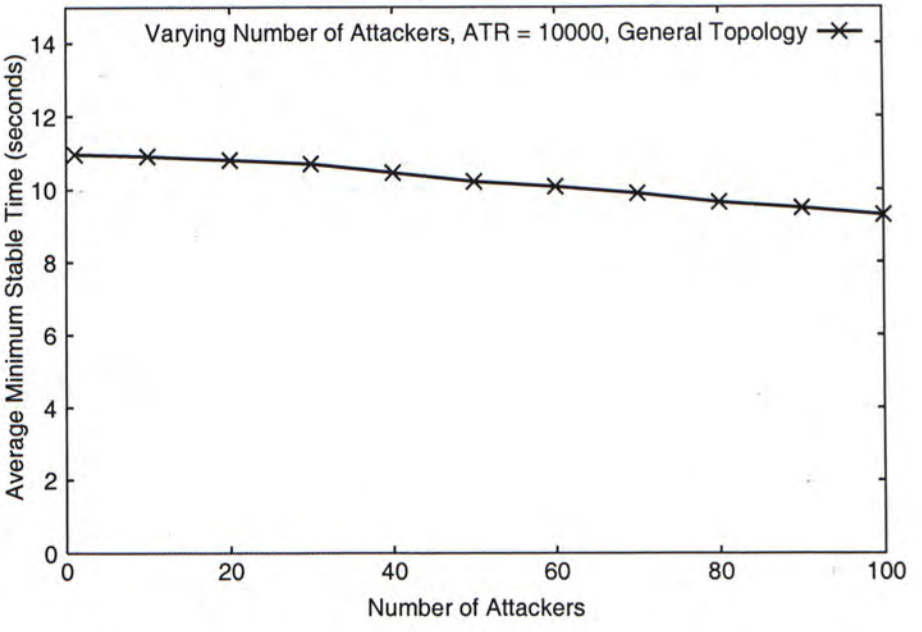

Figure 5.11: Influence on t_{min} by different number of attackers

attackers.

5.4.2 Simulation 4.B - Influence on t_{min} by different number of attackers

In this simulation, the relationship between the number of attackers and the average minimum stable time t_{min} in Internet topology are analyzed. We use the same traceroute dataset as Simulation 4.A. The attackers are randomly assigned to different locations of disjoint attack paths. We evaluate the average minimum stable time t_{min} for different number of attackers. In Figure 5.11, we see that the average minimum stable time t_{min} of our attacker location scheme decreases when the number of attackers increases. Again, this illustrates the effectiveness and robustness of our traceback methodology even under a large-scale DDoS attack.

Chapter 6

Experiments

In this chapter, we perform experiments to illustrate the effectiveness in locating the attackers. In particular, we show the correctness and robustness of using our DDoS traceback methodology to find the minimum stable time t_{min} such that we can compute the intensities of the local traffic rates for all routers in the attack graph $\mathcal{G}$. Note that a smaller value of t_{min} implies that we can find the locations of the attackers earlier. We illustrate various factors which can influence the values of the minimum stable time t_{min}. To test the performance of the our DDoS traceback methodology in real setting, we use 20 Linux routers to setup a DDoS attack testbed shown in Figure 6.1 and 6.2. The Linux routers are running on a Pentium IV machines and we add a kernel module in the routers to support probabilistic edge marking. Since probabilistic edge marking requires only modifying the identification field of the IP header of a packet, it has essentially the same processing complexity as standard IP forwarding, and adds little computational overhead or transmission delay at a deployment router. The well-known attack tool *TFN2K* will be used to launch UDP flood attack, SYN flood attack, and ICMP flood attack. In the experiments, we use a general network topology as depicted in Figure 6.3. To derive the attack graph, the victim $\mathcal{V}$ informs the participating routers to mark packets with probability $p = 1/10$. To estimate whether we have reached the stability conditions specified by Equation (3.6) or (4.9), we

set $p_{threshold} = 95\%$. In other words, the estimation of the local traffic rates is highly accurate. We define a variable, **Attack Traffic Ratio** (ATR), which is the ratio of the attacker's average traffic rate to the normal average local traffic. For example, if the normal average local traffic is 10 pkts/sec and ATR is set to 10, then the attacker's average traffic rate is equal to 100 pkts/sec.

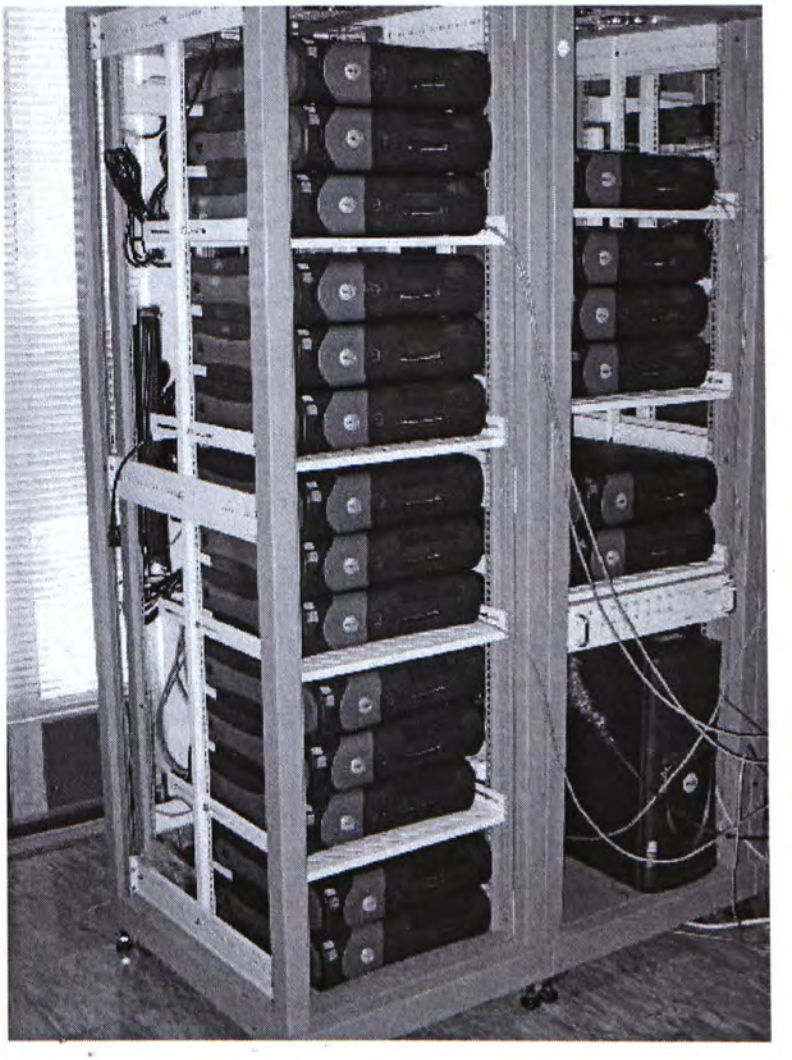

Figure 6.1: The front view of the DDoS attack testbed

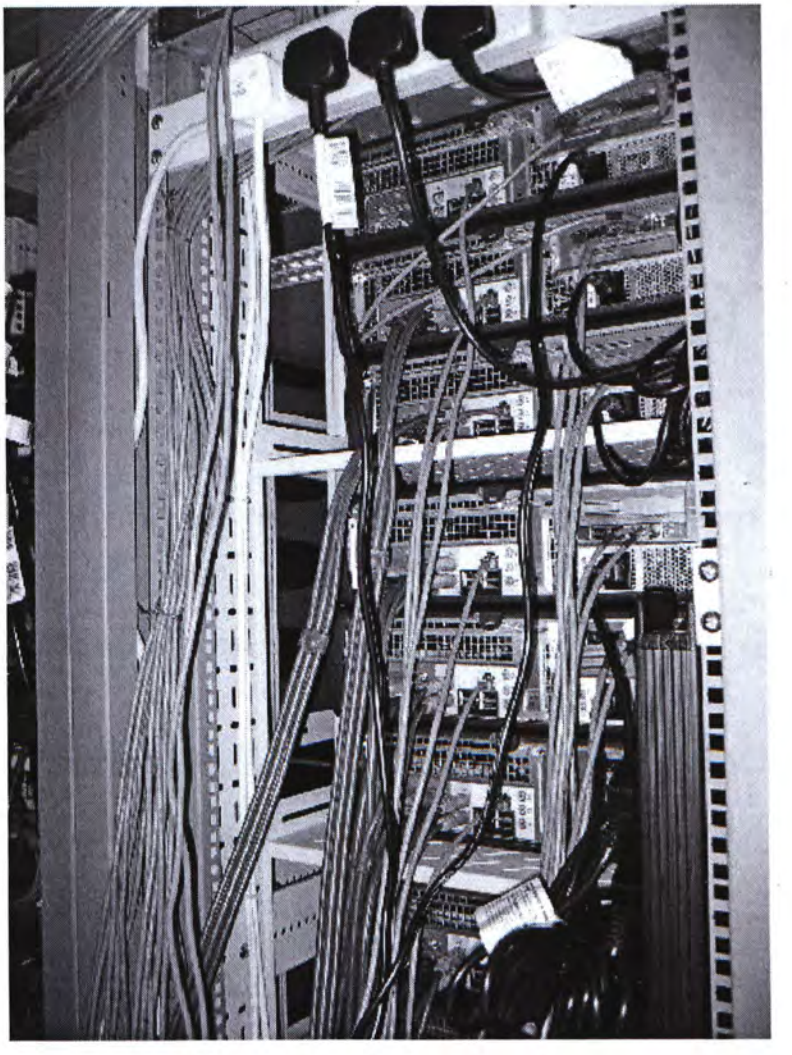

Figure 6.2: The back view of the DDoS attack testbed

6.1 Experiment 1: Simple DoS Attack

This set of experiments evaluates the correctness of the estimation of the minimum stable time t_{min} based on the mathematical model (3.6) presented in Section 3.1. We show that the estimation is robust and accurate for different types of DDoS attacks, for example, UDP flood attack, SYN flood attack, and ICMP flood attack. For this set of experiments, we maintain the normal average local traffic to 10 pkts/sec. The ATR is set to 10 (in other words, the

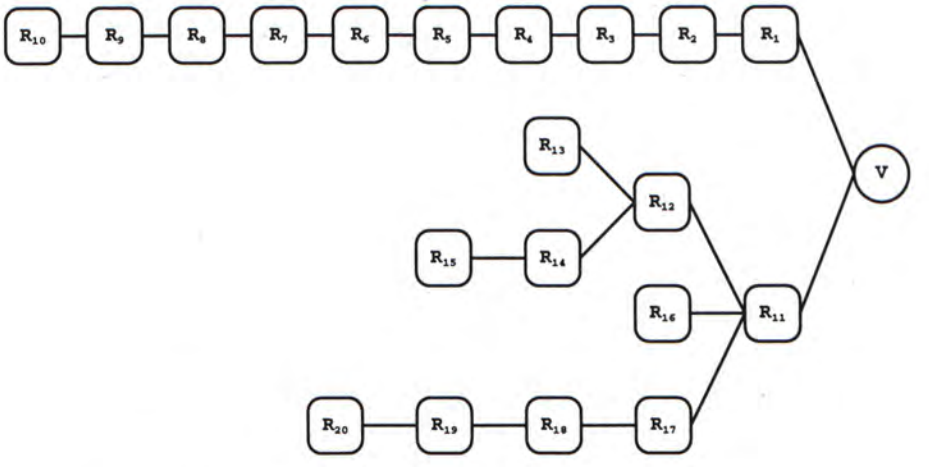

Figure 6.3: General network topology with 20 routers

attacker's average traffic rate is 100 pkts/sec).

6.1.1 Experiment 1.A - Influence on t_{min} by different types of DDoS attack

In this experiment, we use a general network topology in Figure 6.3 with 20 routers. There is one attacker and he is located at the furthest router R_{10}. The three types of attack traffic are generated: (1) UDP flood (i.e., large number of UDP packets are sent to the victim), (2) SYN flood (i.e., large number of TCP packets with SYN-bit set are sent to the victim), (3) ICMP flood (i.e., large number of ICMP packets are sent to the victim) To compute the minimum stable time t_{min}, we use the mathematical derivation in Equation (3.6). Table 6.1 illustrates that with different values of t_{min}, our theoretical computed $p_{threshold}$ value is very close to the experimental $p_{threshold}$ values for different types of DDoS attacks. This indicates that our methodology is very accurate and robust in estimating the minimum stable time t_{min} and, regardless of the type of DDoS attacks, it takes around 30 seconds to determine the local traffic rates of all routers in the attack graph $\mathcal{G}$.

t_{min} (sec)	$p_{threshold}$ (%)			
	Theoretical Computed Values	**Experimental Values**		
		UDP Flood	**SYN Flood**	**ICMP Flood**
5.0	76.85	77.03	76.91	76.89
10.0	83.73	83.95	83.77	83.81
15.0	88.14	88.20	88.05	88.41
20.0	91.18	90.93	90.98	91.24
25.0	93.36	93.20	93.42	93.33
30.0	94.96	94.72	94.87	95.03

Table 6.1: Minimum Stable Time: Theoretical vs. Experimental Results for different types of DDoS attack

6.1.2 Experiment 1.B - Influence on t_{min} by different length of the attack path

In this experiment, the relationship between the length of an attack path and the minimum stable time t_{min} is analyzed. We use a general network topology with varying length of attack path and compute the minimum stable time t_{min}. We assume there is one attacker and he is located at the furthest router. Figure 6.4 shows that as the length of the attack path increases, the minimum stable time t_{min} also increases. The reason for this linear growth is that the victim site $\mathcal{V}$ needs to take a longer time to collect sufficient number of *marked* packets from the furthest router.

6.2 Experiment 2: Coordinated DoS Attack

In the previous set of simulations, we analyze the performance of the attacker location scheme by assuming there is only one attacker. In this set of simulations, we extend the performance study to the coordinated DoS attack. We evaluate the minimum stable time t_{min} of the attacker location scheme based

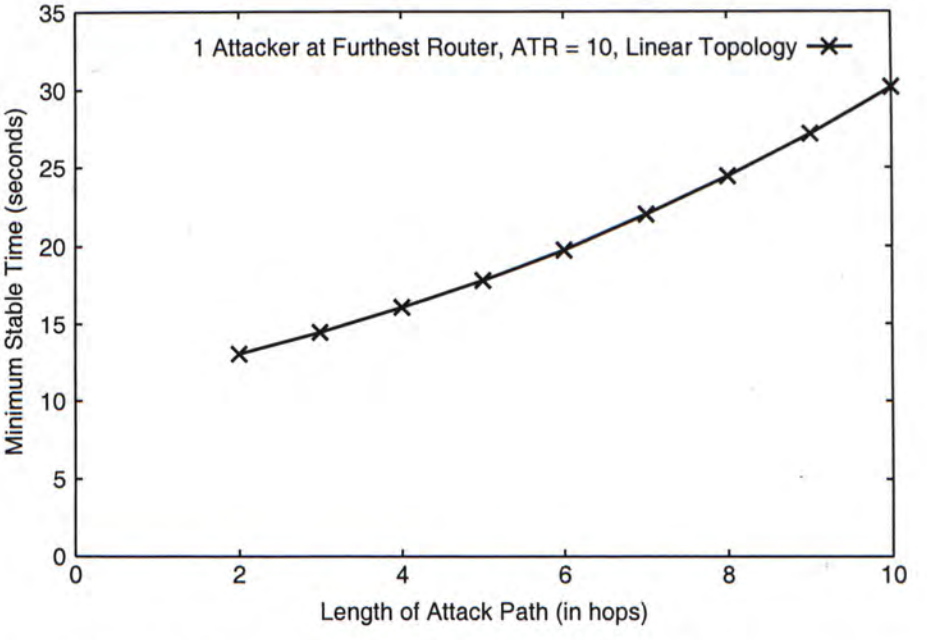

Figure 6.4: Influence on t_{min} by different length of the attack path

on the mathematical model (4.9) presented in Chapter 4. For this set of experiments, we maintain normal average local traffic to 10 pkts/sec and vary the attackers' average traffic rates to 50, 300, 400, and 500 pkts/sec. (or ATR = 5, 30, 40, 50).

6.2.1 Experiment 2.A - Influence on t_{min} by the relative positions of the attackers

In this experiment, we study how the number of attackers and their relative positions influence the value of the minimum stable time t_{min}. We consider three attackers in a general network topology and these three attackers are distributed by three different configurations. For configuration I, these three attackers are located at routers R_1, R_2 and R_3 (e.g., the 3 closest routers to the victim $\mathcal{V}$). For configuration II, these three attackers are located at the three furthest routers from the victim $\mathcal{V}$. In configuration III, these three attackers

are *evenly distributed* along the attack path, for example, they are located at routers $R_{\lfloor \frac{d*i}{3} \rfloor}$ where d is the length of attack path and i=1,2,3. Figure 6.5 illustrates the minimum stable time t_{min} when ATR= 5 and $p_{threshold}$ = 95%. We observe that when the attackers are closer to the victim $\mathcal{V}$ (e.g., configuration I), the achieved minimum stable time t_{min} is lower than other configurations. The minimum stable time t_{min} achieves highest value when attackers are evenly distributed in the network (e.g., configuration III). The reason for this ordering is that the estimation of local traffic rate at the furthest router takes longer time to become stable while the estimation of local traffic rates at the nearest router takes less time. Regardless of the distribution of the attackers' positions, we can determine their locations within 30 seconds, which shows the effectiveness of the proposed methodology.

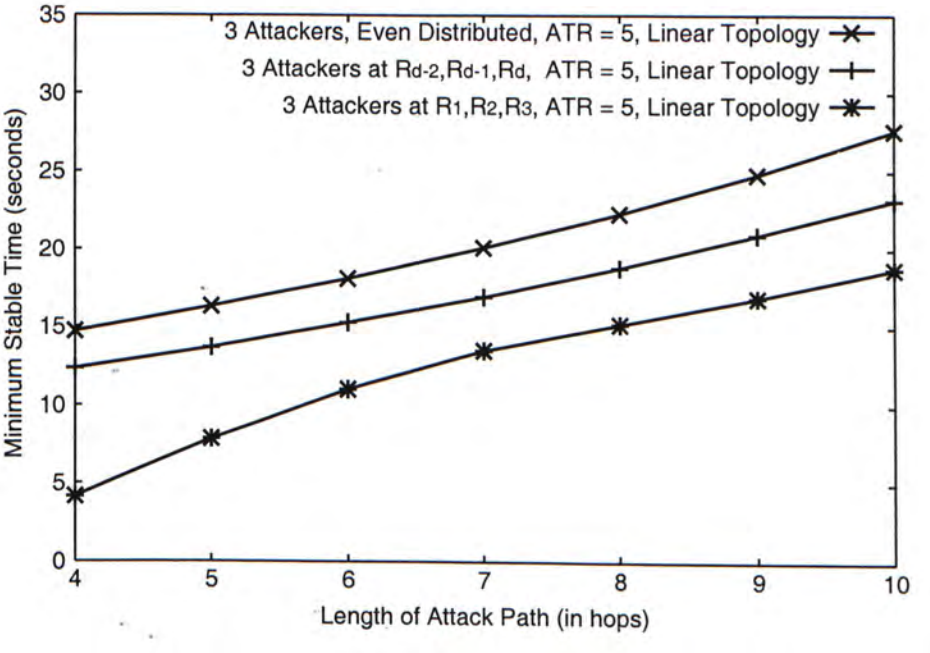

Figure 6.5: Influence on t_{min} by the relative positions of the attackers

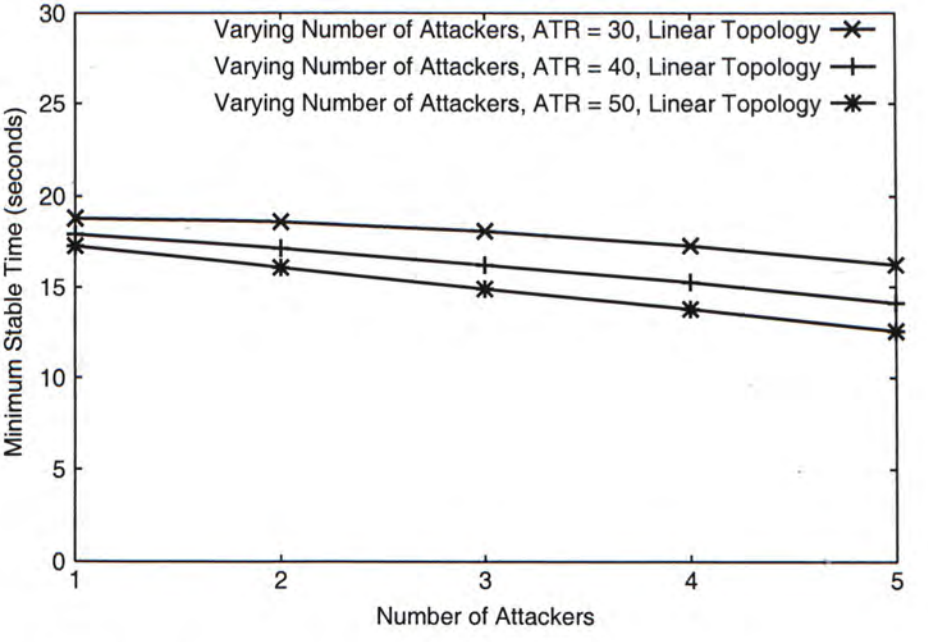

Figure 6.6: Influence on t_{min} by different number of attackers

6.2.2 Experiment 2.B - Influence on t_{min} by different number of attackers

In this experiment, we vary the number of attackers in the attack graph and evaluate the average minimum stable time t_{min} under different attackers' traffic rates. The attackers are randomly assigned to different locations. Figure 6.6 shows that as the number of attackers increases, the average minimum stable time t_{min} decreases. The reason for the decrease of t_{min} is that larger number of attackers, more marked packets will be collected , faster to satisfy the stability conditions of the local traffic estimation. This illustrates the effectiveness and robustness of our traceback methodology even under a large-scale DDoS attack.

Chapter 7

Related Work

One major security problem of the IP protocol is that the source address can be filled by any user [Bel]. There is no provision to discover the true origin of the packet. Various researchers have proposed different approaches to solve the DDoS attack problem, including ingress filtering [FS98], link testing [BC00], logging [SPS+01], and even throttle technique [YLL02].

While ingress filtering can reduce the number of attack packets, the best way to address the distributed denial-of-service attack is to traceback the attackers and stop their attacks. One of the traceback approach is called the link testing. Link testing is to interactively test its upstream link starting from the nearest routers. This process is repeated recursively until the source of attack is found. There are many ways to implement the link testing and we explain two of them: input debugging and controlled flooding. For input debugging, the victim site $\mathcal{V}$ develops attack signature based on the characteristics of the attack packets. Then victim site $\mathcal{V}$ requests network administrators to filter attack packets on some egress ports and identifies which ingress ports they arrived from. The procedure is repeated recursively and the origin of the attack can be found. This approach is usually implemented manually and Stone [Sto00] proposed automatic way to trace the attackers within their own networks. But the communication overhead for exchanging message is very high and operations are constrained by different network administrations. Burch

and Cheswick [BC00] proposed another approach, which is called the controlled flooding. Victim site $\mathcal{V}$ interactively floods its upstream links starting from the nearest routers. Then the victim site $\mathcal{V}$ observes the changes in the rate of attack packets and identifies which link the attack packets came from. Selective flooding is repeated to further routers until the full attack path is found. Controlled flooding consumes many network resource such as network bandwidth. If it is not handled or implemented carefully, this approach can be considered as another form of attack.

Both link testing approaches require the attack remains active during traceback process. Logging is another approach to traceback even after the attack has completed. Partial packet information will be stored in routers for the traversed packets and this information provides a trail of the attack path. Full attack path can be reconstructed by applying some data extraction methods. The advantage of this approach is that it does not increase the volume of traffic, but it increases the storage requirement of the participating routers. Snoeren *et al.* [SPS+01] proposed efficient hash-based technique to store the packet digestion, and stated the storage requirement is approximately below one percentage of the link capacity per unit time. However, because tens of thousands of packets can traverse a router each second, the logging data can still grow quickly to an enormous size and this is especially true for a very high speed link. Therefore, the storage overhead of routers would still be a problem.

While logging requires huge amount of space in each router, Bellovin [SMB] proposed ICMP traceback (and later extensions by Wu *et al.* [WZMM, MMW+01]) which can traceback the attackers without incurring much overhead on the routers. The routers probabilistically generate an authenticated copy of a packet, including information about the adjacent routers along the path to the destination. These information can be used to reconstruct the path to the attackers. However, ICMP traffic is different from the normal traffic and may

be blocked or rate limited. It also needs key distribution management to deal with the faking ICMP packet problem.

Savage *et al.* [SWKA00] proposed probabilistic marking for traceback without generating separate ICMP packets to the victim site $\mathcal{V}$. Routers mark packets probabilistically and store the edge information in the IP header. Each piece of information represents a sample edge of the attack graph. Victim site $\mathcal{V}$ collects the marked packets to reconstruct the attack graph. This approach does not need the coordination among the network administrators and it does not increase the traffic flow or the storage requirement of a router. Lee and Park [PL01] analyzed this marking approach and pointed out that spoofing of the marking field may impede traceback by the victim site $\mathcal{V}$. Attackers may choose the spoofed marking value, source address to hide themselves. Dean *et al.* [DFS01] formulated the traceback problem as a polynomial reconstruction problem. They used algebraic coding theory to encode traceback information in the packet, similar to Savage's approach [SWKA00]. It also suffers the same spoofing problem and may be more vulnerable without the distance field in the marking. Song and Perrig [SP01] reported that if the victim site $\mathcal{V}$ knows the map of its upstream routers, it does not need the full IP address in the packet marking. They improved Savage's marking approach [SWKA00] by hashing so as to achieve a lower false positive rate and a lower computation overhead. They proposed efficient authentication of packet markings to filter packets with spoofed markings from the attackers. Note that approaches by Savage [SWKA00] and Song [SP01] provide the topology of the attack graph only. Our approach can be viewed as a complementary approach to them so as to locate the potential attackers in the attack graph.

Chapter 8

Conclusion

In this thesis, we consider the traceback problem during a DDoS attack. Instead of dealing with the issue of *detecting* a DDoS attack, we address how we can locate and eliminate potential attackers. Our approach uses the probabilistic marking algorithm: a victim site $\mathcal{V}$, upon discovering that it is being attacked, will request a set of routers to mark all packets targeted to $\mathcal{V}$ with certain probability p. Based on the collected marked packets, the victim site $\mathcal{V}$ can then construct an attack graph. We enhance the probabilistic marking algorithm by determining the local traffic rate of each router in the attack graph. Based on the traffic intensities, we can send signal to the corresponding routers, (e.g., those routers which consume highest percentages of the victim's resource) to eliminate their local traffics. One important technical contribution we made is that we provide a theoretical approach to determine the minimum stable time t_{min}, which is the minimum time it takes to *accurately* determine the local traffic rate of every participating router in the attack graph so that we can locate the potential attackers. We carried out simulations and experiments to illustrate the effectiveness and robustness of our algorithm under linear or general network topology. We also show the robustness of our estimation when the packet arrival process takes different forms, for example, Poisson, constant, bursty and MMPP. Experimental results show that we can locate the attackers within a short duration. We believe this is a valuable first

step towards automating a wide area network traceback facility.

Bibliography

[BC00] Hal Burch and Bill Cheswick. Tracing anonymous packets to their approximate source. In *Usenix LISA*, December 2000.

[Bel] S.M. Bellowin. Security problems in the TCP/IP protocol suite. pages 32 – 48.

[BF88] Richard L. Burden and J. Douglas Faires. *Numerical Analysis.* PWS-Kent Publishing Company, Boston, 1988.

[ddo00] "computer emergency response team, cert advisory ca-2000-01: Denial-of-service developments". http://www.cert.org/advisories/ ca-2000-01.html. 2000.

[DFS01] Drew Dean, Matt Franklin, and Adam Stubblefield. An algebraic approach to IP traceback. In *Proceedings of Network and Distributed System Security Symposium, NDSS '01*, February 2001.

[FS98] P. Ferguson and D. Senie. Rfc 2267: Network ingress filtering: Defeating denial of service attacks which employ IP source address spoofing. *The Internet Society*, January 1998.

[How98] J.D. Howard. An analysis of security incidents on the internet. *PhD Thesis, Carnegie Mellon University*, August 1998.

[MMW+01] Allison Mankin, Dan Massey, Chien-Long Wu, S. Felix Wu, and Lixia Zhang. On design and evaluation of intention-driven ICMP

traceback. In *Proceedings of IEEE International Conference on Computer Communications and Networks*, 2001.

[Nel95] Randolph Nelson. *Probability, Stochastic Processes, and Queueing Theory: The Mathematics of Computer Performance Modeling.* Springer-Verlag, 1995.

[PL01] Kihong Park and Heejo Lee. On the effectiveness of probabilistic packet marking for IP traceback under denial of service attack. In *Proceedings of IEEE INFOCOM '01*, pages 338 – 347, 2001.

[Ros96] Sheldon M. Ross. *Stochastic Processes.* John Wiley Series, New York, 1996.

[SMB] Editor Steven M. Bellovin. ICMP traceback messages, internet draft: draft-bellovin-itrace-00.txt. submitted Mar. 2000, expiration date Sep. 2000.

[SP01] Dawn X. Song and Adrian Perrig. Advanced and authenticated marking schemes for IP traceback. In *Proceedings of IEEE INFOCOM '01*, April 2001.

[SPS+01] Alex C. Snoeren, Craig Partridge, Luis A. Sanchez, Christine E. Jones, Fabrice Tchakountio, Stephen T. Kent, and W. Timothy Strayer. Hash-based IP traceback. In *Proceedings of the ACM SIGCOMM 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, August 2001.

[Sto00] Robert Stone. Centertrack: An IP overlay network for tracking dos floods. In *Proceedings of 9th USENIX Security Symposium*, August 2000.

[SWKA00] Stefan Savage, David Wetherall, Anna Karlin, and Tom Anderson. Practical network support for IP traceback. In *Proceedings of the 2000 ACM SIGCOMM Conference*, pages 295 – 306, Stockholm, Sweden, August 2000.

[tra99] Internet scanning database, http://cm.bell-labs.com/who/ches/map/dbs/index.html. 1999.

[WZMM] S. Felix Wu, Lixia Zhang, Dan Massey, and Allison Mankin. Intention-driven ICMP trace-back, internet draft: draft-wu-itrace-intention-00.txt. submission date Feb. 2001, expiration date Aug. 2001.

[YLL02] David K. Y. Yau, John C. S. Lui, and Feng Liang. Defending against distributed denial-of-service attacks with max-min fair server-centric router throttles. In *IEEE International Workshop on Quality of Service (IWQoS)*, May 2002.